SIMATIC Modbus/TCP

Communication via the integrated PN interface of the CPU

Programming manual · 11/2017

SIMATIC

Answers for industry.

SIEMENS

SIEMENS

SIMATIC

SIMATIC Modbus/TCP Communication via the integrated PN interface of the CPU

Programming Manual

11/2017

A5E39465006-AB

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

A5E39465006-AB
Ⓟ 11/2017 Subject to change

Preface

Purpose of this manual

With the information in this manual, you can set up and commission a link between a CPU with integrated PN interface and a device that supports the Modbus/TCP protocol.

Contents of the manual

This manual describes the function of the Modbus function block and its parameter assignment.

The manual covers the following topics:

- Product description
- Getting Started
- Commissioning
- Parameter assignment of the Modbus communication
- Licensing
- MODBUSPN function block
- Additional blocks
- Diagnostics
- Example of the application

Validity of the manual

This manual is valid for the following software:

Product	Identification number	as of version
MODBUS/TCP PN CPU	6AV6676-6MB20-3AX0 6AV6676-6MB20-3AD0	3.0
FB 900 "MODBUSPN"		4.0
FB 901 "MOD_CLI"		2.0
FB 903 "MOD_SERV"		2.0

Note

This manual contains the description of the FB valid at the time the manual was published.

Additional sources of information

You can find all other information relating to the PN-CPUs and the IM 151-8 PN/DP CPU (installation, commissioning etc.) in the manuals:

SIEMENS
SIMATIC S7-300
CPU 31xC and CPU 31x: Installation
Operating instructions
A5E00105491-07

SIEMENS
SIMATIC S7-300
CPU 31xC and CPU 31x, Technical specifications
Manual
A5E00105474-07

SIEMENS
SIMATIC S7-400
Automation System S7-400: Installation
Operating instructions
A5E00850740-01

SIEMENS
SIMATIC S7-400
Automation System S7-400 CPU data
Manual
A5E00850745-06

SIEMENS
SIMATIC
Distributed I/O ET 200S
Interface module IM151-8 PN/DP CPU
Operating instructions
A5E02049033-01

SIEMENS
Product Information on
CPU314C-2 PN/DP, 6ES7314-6EH04-0AB0
CPU315-2 PN/DP, 6ES7315-2EH13-0AB0
CPU315F-2 PN/DP, 6ES7315-2FH13-0AB0
CPU317-2 PN/DP, 6ES7317-2EK13-0AB0
CPU317F-2 PN/DP, 6ES7317-2FK13-0AB0
CPU317-2 DP, 6ES7317-2AJ10-0AB0
CPU317F-2 DP, 6ES7317-6FF03-0AB0
CPU319-3 PN/DP, 6ES7318-3EL00-0AB0
CPU319F-3 PN/DP, 6ES7318-3FL00-0AB0
A5E01103134-03

You can find additional information relating to STEP 7 in the following manuals:

SIEMENS SIMATIC software
Basic software for S7 and M7
STEP 7 user manual
C79000-G7000-C502-..

SIEMENS SIMATIC Software
System Software for S7-300/400
System and Standard Functions
Reference Manual
C79000-G7000-C503-02

Queries

Your Siemens contact from whom you received this function block will be pleased to provide answers to any open issue relating to the use of the FBs described in this manual.

Conventions

The designation PN-CPU is used below in this documentation. The details are valid for the **PN-CPU**s of series **314C**, **315**, **317**, **319**, **412**, **414** and **416** as well as for the **IM 151-8 PN/DP CPU**.

Area of application

The function blocks described in this manual set up a connection between a PN-CPU and Modbus devices of third-party manufacturers.

Table of contents

1 Product description

1.1 Possible applications

Integration in the system environment

This function block instruction represents a software product for CPUs with integrated PN interface of the SIMATIC S7-300, S7-400 and IM 151-8 PN/DP CPU.

Function of the FBs

These function blocks enable communication between an S7 CPU with integrated PN interface and a device which supports the Modbus/TCP protocol.

Data transmission takes place according to the client-server principle.

The SIMATIC S7 can be operated as client as well as server during the transfer.

Use of the port number 502

The Modbus/TCP protocol usually runs via port 502. This port number is only available for PN CPUs with the corresponding firmware version. Information regarding enabling of port numbers is available here (http://support.automation.siemens.com/WW/view/en/34010717).

Specific CPU types can maintain and operate connections to multiple clients simultaneously via the local port 502. The technical details of this topic are explained in the section "Multiple connections at port 502".

1.2 Hardware and software requirements

Usable modules for MODBUSPN

You will find current hardware requirements here (http://support.automation.siemens.com/WW/view/en/104946406).

Software versions

The use of the MODBUSPN FB is possible as of **STEP 7, Version 5.5**.

Memory requirements

The MODBUSPN FB requires approx. 8 KB of work memory and 9 KB of load memory.

The MOD_CLI FB requires 9 KB work memory and load memory.

The MOD_SERV FB requires 9 KB work memory and load memory.

You will find the precise lengths of the blocks in their properties in the SIMATIC Manager.

2 Getting Started

Procedure

1. Installation of "SIMATIC Modbus/TCP PN CPU" and inclusion of the MODBUS blocks in the user project

   => Section 3.1 (Page 15)

2. Setting the parameter DB MODBUS_PARAM according to requirements (IP address, port number, client/server, connection setup upon restart, Modbus register, DB areas, etc.)

   => Section 4 (Page 19)

3. Calling and parameter assignment of the MODBUSPN Modbus block in the required OB

   => Sections 6.1 (Page 35) and 6.2 (Page 38)

4. Loading the user program to the CPU and licensing of the Modbus block for this CPU.

   => Section 5 (Page 31)

3 Commissioning

General information

The following information on STEP 7 and on configuring the communication connections relates to STEP 7, Version 5.5.

In later versions, sequences, name and directory information may have changed.

Requirements

STEP 7 basic knowledge, STL knowledge, PLC basic knowledge

3.1 Installing the library on the STEP 7 PG/PC

Scope of delivery

The supplied CD contains a setup with which the library "Modbus_PN_CPU", the sample projects and the manuals in English and German can be installed in the relevant STEP 7 directories.

The CD also contains the manuals in PDF format.

Requirements

To be able to perform the installation, the STEP 7, V5.5 configuration software must first be installed.

Installation

Insert the Modbus CD in the CD-ROM drive of your PG/PC. If the setup program does not start automatically, install as follows:

1. Select the CD-ROM drive in the Windows Explorer, open the Setup directory and start the setup program.
2. Follow the instructions displayed by the installation program step by step.

You will now find

- the library in "\Program Files\Siemens\Step7\S7libs"
- the sample projects in "\Program Files\Siemens\Step7\Examples"
- the manual in "\Program Files\Siemens\Step7\S7manual\S7Comm"
- the software registration form in "\Program Files\Siemens\Step7\S7LIBS\Modbus_PN_CPU".

The manual can also be opened using the shortcut under "\Program Files\Siemens\Documentation".

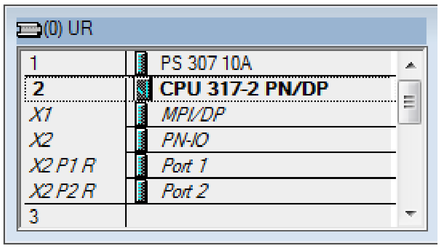
3.2 CPU - Assign IP address

In this example of the assignment of the IP address, a CPU 317-2PN/DP is used.

Procedure

Before you can carry out the configuration, you must have created an S7 project with STEP 7.

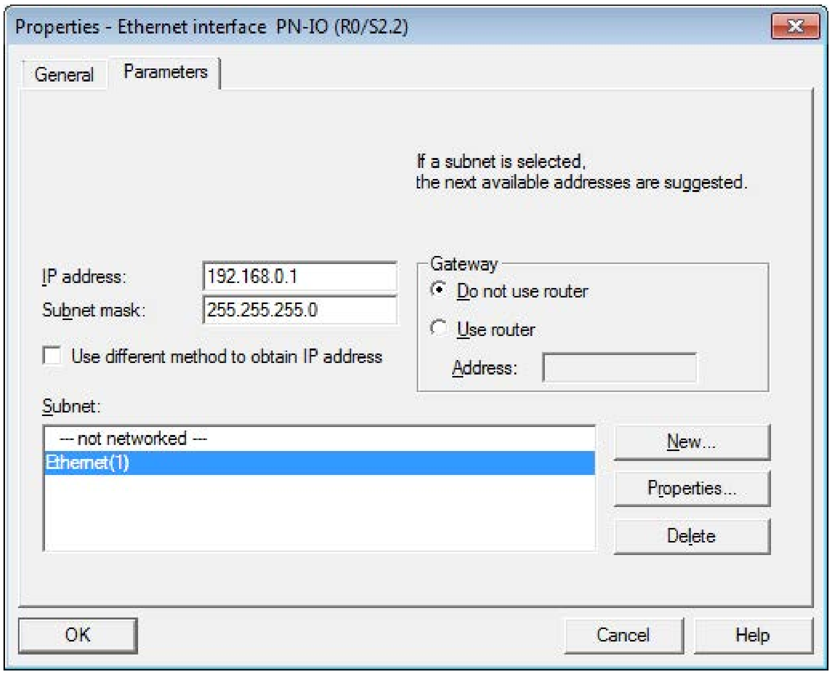
1. Open HWConfig and insert a CPU 317-2 PN/DP at slot 2.



(0) UR

1	PS 307 10A
2	**CPU 317-2 PN/DP**
X1	*MPI/DP*
X2	*PN-IO*
X2 P1 R	*Port 1*
X2 P2 R	*Port 2*
3	

2. Double clicking on the line X2 opens the properties dialog of the PN-IO interface.



Properties - Ethernet interface PN-IO (R0/S2.2)

If a subnet is selected,
the next available addresses are suggested.

IP address: 192.168.0.1
Subnet mask: 255.255.255.0
☐ Use different method to obtain IP address

Gateway
[✓ Do not use router]
Use router
Address:

Subnet:
--- not networked ---
Ethernet(1)

3. Enter the IP address and the subnet mask. If you set up a connection via a router, you also have to enter the address of the router.
4. Click the "New..." button and assign a name for a new Industrial Ethernet subnet. Then, click "OK".

   Result: You have created a new Industrial Ethernet subnet.
5. Click "OK".

   Result: The property window of the PN-IO interface of the CPU 317-2 PN/DP is closed.

3.3 Inserting the function blocks in the program

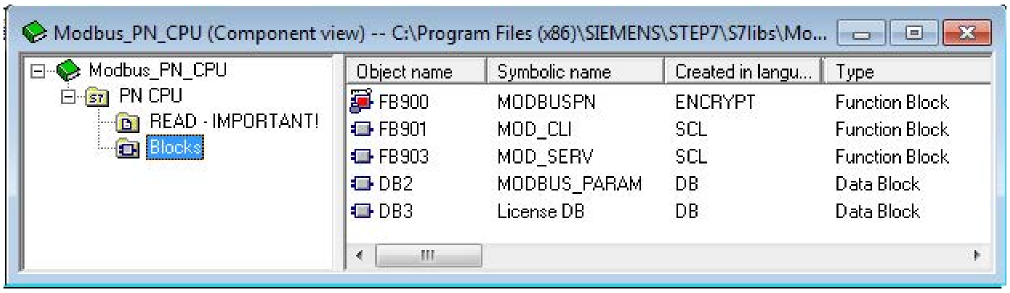
Content of the Modbus library:

The following FBs are required for Modbus communication:

- MODBUSPN
- MOD_CLI
- MOD_SERV

To be able to insert these in your project, you need to copy the blocks from the library.

You will also find the parameter data block MODBUS_PARAM and the license DB as templates in the library. To simplify your work, you can also copy these to your project.


Modbus_PN_CPU (Component view) -- C:\Program Files (x86)\SIEMENS\STEP7\S7libs\Mo...
Modbus_PN_CPU
PN CPU
READ - IMPORTANT!
Blocks

Object name	Symbolic name	Created in langu...	Type
FB900	MODBUSPN	ENCRYPT	Function Block
FB901	MOD_CLI	SCL	Function Block
FB903	MOD_SERV	SCL	Function Block
DB2	MODBUS_PARAM	DB	Data Block
DB3	License DB	DB	Data Block

Blocks of the standard library

The following FBs are required for Modbus communication:

- TSEND (FB63)
- TRCV (FB64)
- TCON (FB65)
- TDISCON (FB66)

These communication blocks are contained in the "Standard Library - Communication Blocks" library and must also be included in your project.

Note

Note that the following versions of the FBs are requirements for the proper operation of the MODBUSPN FB:

- TSEND V2.1
- TRCV V2.2
- TCON V2.4
- TDISCON V2.1

3.4 Multiple connections to port 502

Some CPUs can multiplex TCP connections. This means that several MODBUS clients can connect to port 502 of the CPU (multiport). The CPU acts as MODBUS server.

Here (http://support.automation.siemens.com/WW/view/en/34010717) you will find information about which CPU allows multiple use of the port 502 and with which firmware version.

Requirements

To be able to use this function, the following settings need to be made during parameter assignment:

- CP is server
- Port 502 as local port
- Passive connection establishment

Number of enabled connections

The number of connections that a CPU can accept on port 502 depends on the device and is available in the technical specifications of the CPU.

Configuration

For each client that wants to connect to port 502 of the server, one unique connection is required in the parameter DB and one Modbus block instance each in the program.

Parameter assignment of the Modbus communication 4

4.1 Overview

General information

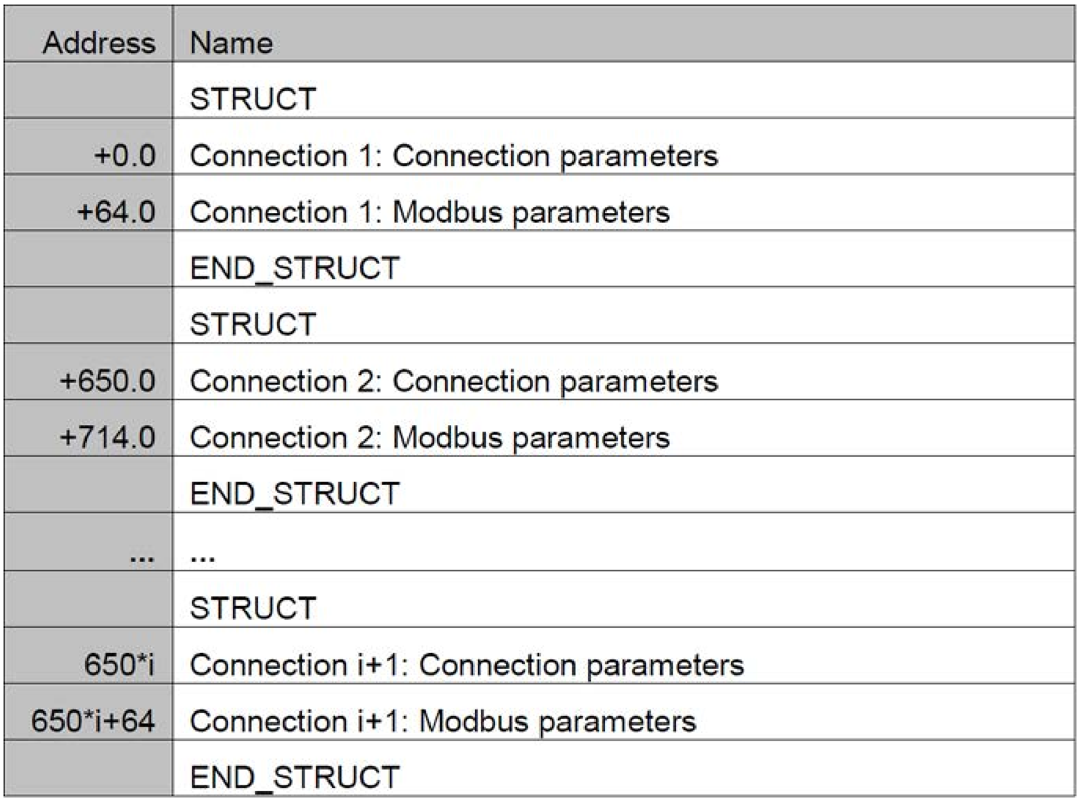
You do not need to configure a connection in NetPro for communication using the integrated PN interface of the CPU. The connections are set up and terminated using the TCON (FB65) and TDISCON (FB66) function blocks.

Several connections to different communications partners can be configured and established at the same time. The number of simultaneously established connections depends on the CPU.

MODBUS_PARAM connection data block

The data required to set up the connections and to process the Modbus frames is defined in a structure – the MODBUS_PARAM parameter data block. The connection-specific data is first saved. The connection-specific data is followed by the Modbus parameters.

The parameter data block can contain the configuration data of all connections. You can also create a separate parameter data block for each connection. A completed structure is included in the "Modbus_PN_CPU" library as an example.



Address	Name
	STRUCT
+0.0	Connection 1: Connection parameters
+64.0	Connection 1: Modbus parameters
	END_STRUCT
	STRUCT
+650.0	Connection 2: Connection parameters
+714.0	Connection 2: Modbus parameters
	END_STRUCT
...	...
	STRUCT
650*i	Connection i+1: Connection parameters
650*i+64	Connection i+1: Modbus parameters
	END_STRUCT

Figure 4-1 Param_DB

Connection parameters

The connection-specific parameters, such as the local hardware interface and the IP address of the communication partner, are defined in the connection block. The TCON and TDISCON functions can be used to establish or terminate a connection using these parameters. The exact structure is available in section 4.2.

You have to strictly adhere to the data structure of the connection parameter block because the connection cannot be established otherwise.

Modbus parameters

The data required for operating mode and address reference is stored in the Modbus parameters including, for example, the Modbus areas which are mapped in the data blocks and the operating mode of the S7 as Modbus server or Modbus client. You must adhere to the data structure of the Modbus parameters because they cannot be processed correctly otherwise.

Configuration options

You have two options for configuring the connection and Modbus parameters. On the one hand, the entries can be made with a wizard, making the parameter assignment extremely convenient. On the other hand, the parameters can be set by editing the structure in the parameter data block.

These two options are described in the following sections 4.2 and 4.3.

4.2 Parameter assignment with the wizard

General information

The "Modbus/TCP Wizard" allows convenient configuration of the connections and the Modbus parameters in the MODBUS_PARAM parameter data block. The entire data block is created (connection parameters and Modbus parameters).

We recommend using the wizard for the parameter assignment of MODBUS_PARAM

You can find the wizard here (http://support.automation.siemens.com/WW/view/en/60735352).

4.3 Manual parameter assignment

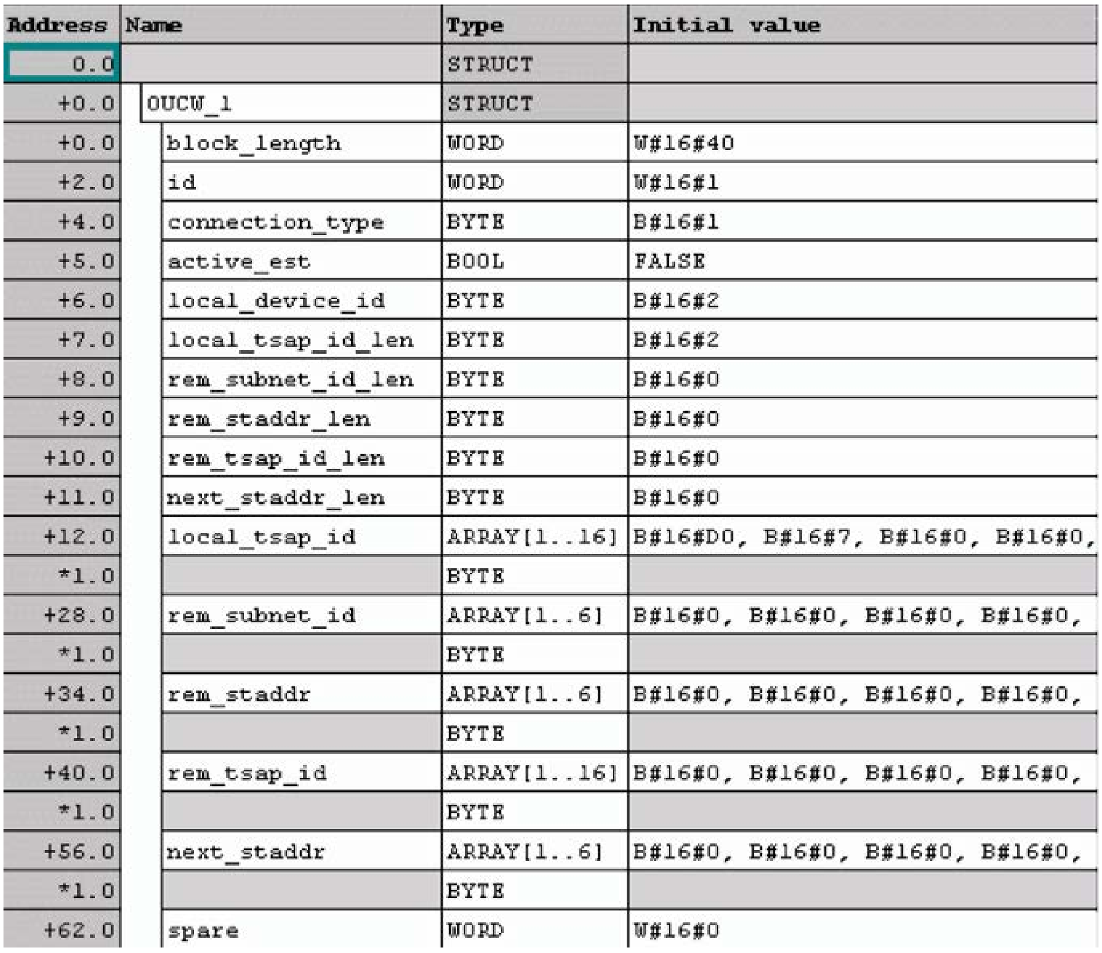
Procedure

Copy DB2 from the "Modbus_PN_CPU" library and insert it in your project. If the number is already being used elsewhere, the DB can be renamed.

The parameters in the MODBUS_PARAM block must not be changed during runtime. After a change of the parameters the Modbus block must be reinitialized with the "Init" parameter.

Structure and adaptations of the connection parameters

A block is required for each connection.



Address	Name	Type	Initial value
0.0		STRUCT	
+0.0	OUCW_1	STRUCT	
+0.0	block_length	WORD	W#16#40
+2.0	id	WORD	W#16#1
+4.0	connection_type	BYTE	B#16#1
+5.0	active_est	BOOL	FALSE
+6.0	local_device_id	BYTE	B#16#2
+7.0	local_tsap_id_len	BYTE	B#16#2
+8.0	rem_subnet_id_len	BYTE	B#16#0
+9.0	rem_staddr_len	BYTE	B#16#0
+10.0	rem_tsap_id_len	BYTE	B#16#0
+11.0	next_staddr_len	BYTE	B#16#0
+12.0	local_tsap_id	ARRAY[1..16]	B#16#D0, B#16#7, B#16#0, B#16#0,
*1.0		BYTE	
+28.0	rem_subnet_id	ARRAY[1..6]	B#16#0, B#16#0, B#16#0, B#16#0,
*1.0		BYTE	
+34.0	rem_staddr	ARRAY[1..6]	B#16#0, B#16#0, B#16#0, B#16#0,
*1.0		BYTE	
+40.0	rem_tsap_id	ARRAY[1..16]	B#16#0, B#16#0, B#16#0, B#16#0,
*1.0		BYTE	
+56.0	next_staddr	ARRAY[1..6]	B#16#0, B#16#0, B#16#0, B#16#0,
*1.0		BYTE	
+62.0	spare	WORD	W#16#0

block_length

This parameter defines the length of the connection parameters and may not be altered.

Fixed value: W#16#40

id

A new connection ID is assigned for each logical connection. When T communication is used, this must be unique throughout the entire CPU. The ID is specified when the MODBUSPN FB is called and on the internal calls of the T blocks (TCON, TSEND, TRCV and TDISCON).

Value range:
W#16#1 to W#16#FFF

connection_type

The connection type for establishing the connection is defined by the TCON function. The CPU determines which value has to be set.

TCP (compatibility mode): B#16#01 for CPU 315 or 317 <= FW V2.3

TCP: B#16#11 for CPU 315 or 317 > FW V2.4, IM 151-8 PN/DP CPU, CPU314C, CPU319, CPU412, CPU414, CPU416

This information can vary depending on the firmware used.
You can find more information here
(http://support.automation.siemens.com/WW/view/en/24294554).

active_est

This parameter refers to the type of connection establishment, active or passive. The Modbus client is responsible for the active connection establishment and the Modbus server for passive connection establishment.

Active connection establishment: TRUE

Passive connection establishment: FALSE

local_device_id

The local_device_id defines the IE interface of the PN CPU in use. Different settings are required depending on the PN CPU type.

IM 151-8 PN/DP CPU	B#16#1
CPU 314C, 315 or 317	B#16#2
CPU 319	B#16#3
CPU 412, 414, 416	B#16#5

local_tsap_id_len

The length of the parameter local_tsap_id (= local port number) is specified.

Active connection establishment: 0

Passive connection establishment: 2

rem_subnet_id_len

This parameter is currently not in use and must have the value B#16#0.

rem_staddr_len

The length of the rem_staddr parameter, which is the IP address of the communication partner. An IP address is not specified for the partner if communication is to take place via an unspecified connection.

Unspecified connection: B#16#0

Specified connection: B#16#4

rem_tsap_id_len

This parameter defines the length of the rem_tsap_id parameter, the port number of the remote communication partner.

Active connection establishment: 2

Passive connection establishment: 0

next_staddr_len

The length of the next_staddr parameter is defined here.
For PN interface: B#16#0

local_tsap_id

You use this parameter to set the local port number. The type of representation differs depending on the connection_type parameter. The CPU determines the value range. The port number must be unique on the CPU.

Table 4- 1 For connection_type B#16#01

local_tsap_id[1]	low byte of the port no. in hex format
local_tsap_id[2]	high byte of the port no. in hex format
local_tsap_id[3-16]	B#16#00

Table 4- 2 For connection_type B#16#11

local_tsap_id[1]	high byte of the port no. in hex format
local_tsap_id[2]	low byte of the port no. in hex format
local_tsap_id[3-16]	B#16#00

rem_subnet_id

This parameter is currently not in use and must have the value 0.

rem_staddr

The IP address of the remote communication partner is entered in this byte array. No IP address is entered in the case of an unspecified connection. The type of representation differs depending on the connection_type parameter.

Example: IP address 192.168.0.1:

Table 4- 3 For connection_type B#16#01

rem_staddr[1]	B#16#01 (1)
rem_staddr[2]	B#16#00 (0)
rem_staddr[3]	B#16#A8 (168)
rem_staddr[4]	B#16#C0 (192)
rem_staddr[5-6]	B#16#00 (reserved)

Table 4- 4 For connection_type B#16#11

rem_staddr[1]	B#16#C0 (192)
rem_staddr[2]	B#16#A8 (168)
rem_staddr[3]	B#16#00 (0)
rem_staddr[4]	B#16#01 (1)
rem_staddr[5-6]	B#16#00 (reserved)

rem_tsap_id

You use this parameter to set the remote port number. The type of representation differs depending on the connection_type parameter. The CPU determines the value range.

Table 4- 5 For connection_type B#16#01

rem_tsap_id[1]	low byte of the port no. in hex format
rem_tsap_id[2]	high byte of the port no. in hex format
rem_tsap_id[3-16]	B#16#00

Table 4- 6 For connection_type B#16#11

rem_tsap_id[1]	high byte of the port no. in hex format
rem_tsap_id[2]	low byte of the port no. in hex format
rem_tsap_id[3-16]	B#16#00

next_staddr

This parameter defines the rack and slot number of the CP in use. This parameter must be set to 0 when you use the integrated PN interface of the CPU.

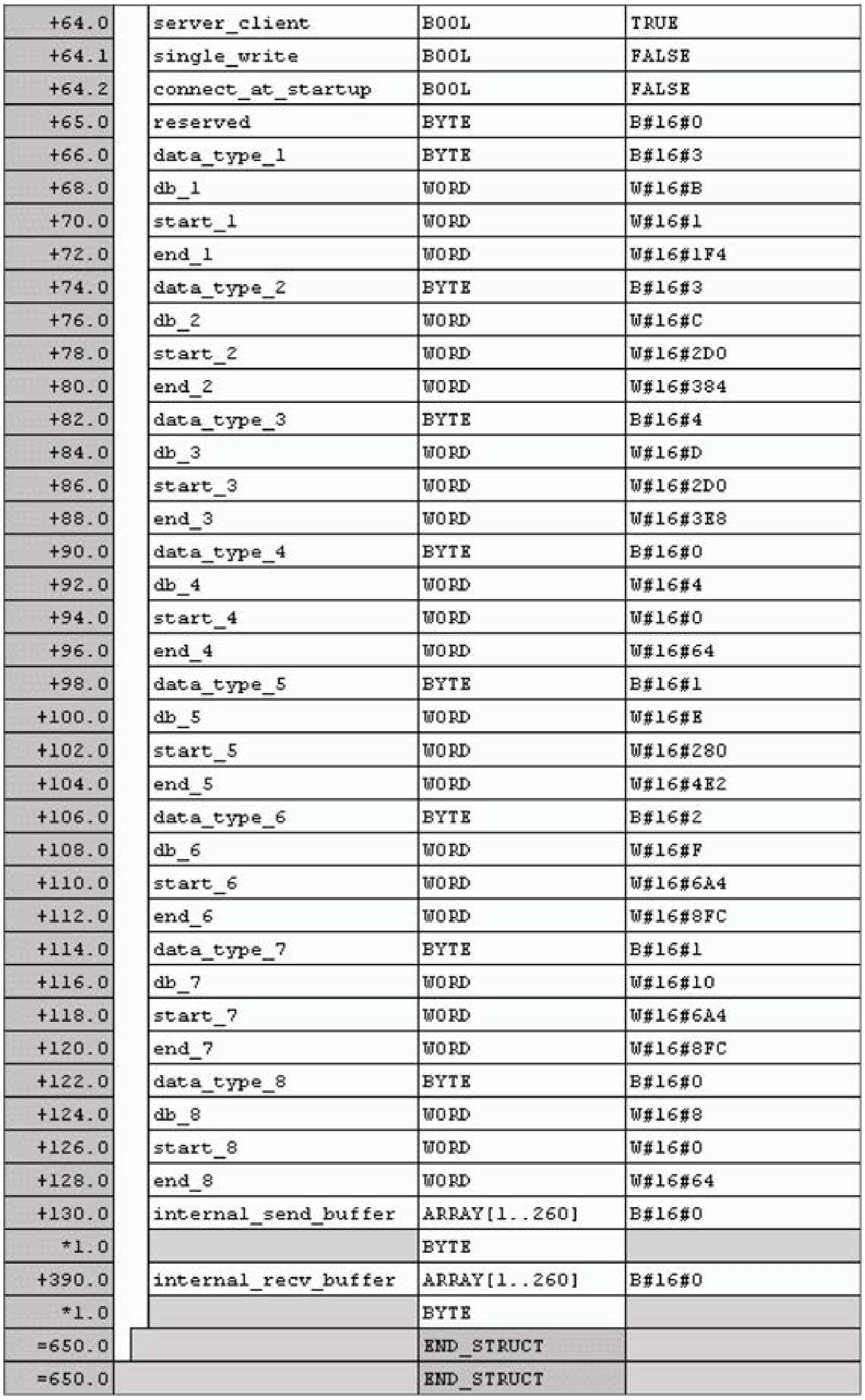
next_staddr[1-6]	B#16#00

spare

This parameter is not in use and must have the default value 0.

Adaptations to the Modbus parameters

The Modbus communication mode and the address mapping of Modbus addresses to the SIMATIC addresses are specified with the Modbus parameters in the MODBUS_PARAM block.



+64.0	server_client	BOOL	TRUE
+64.1	single_write	BOOL	FALSE
+64.2	connect_at_startup	BOOL	FALSE
+65.0	reserved	BYTE	B#16#0
+66.0	data_type_1	BYTE	B#16#3
+68.0	db_1	WORD	W#16#B
+70.0	start_1	WORD	W#16#1
+72.0	end_1	WORD	W#16#1F4
+74.0	data_type_2	BYTE	B#16#3
+76.0	db_2	WORD	W#16#C
+78.0	start_2	WORD	W#16#2D0
+80.0	end_2	WORD	W#16#384
+82.0	data_type_3	BYTE	B#16#4
+84.0	db_3	WORD	W#16#D
+86.0	start_3	WORD	W#16#2D0
+88.0	end_3	WORD	W#16#3E8
+90.0	data_type_4	BYTE	B#16#0
+92.0	db_4	WORD	W#16#4
+94.0	start_4	WORD	W#16#0
+96.0	end_4	WORD	W#16#64
+98.0	data_type_5	BYTE	B#16#1
+100.0	db_5	WORD	W#16#E
+102.0	start_5	WORD	W#16#280
+104.0	end_5	WORD	W#16#4E2
+106.0	data_type_6	BYTE	B#16#2
+108.0	db_6	WORD	W#16#F
+110.0	start_6	WORD	W#16#6A4
+112.0	end_6	WORD	W#16#8FC
+114.0	data_type_7	BYTE	B#16#1
+116.0	db_7	WORD	W#16#10
+118.0	start_7	WORD	W#16#6A4
+120.0	end_7	WORD	W#16#8FC
+122.0	data_type_8	BYTE	B#16#0
+124.0	db_8	WORD	W#16#8
+126.0	start_8	WORD	W#16#0
+128.0	end_8	WORD	W#16#64
+130.0	internal_send_buffer	ARRAY[1..260]	B#16#0
*1.0		BYTE	
+390.0	internal_recv_buffer	ARRAY[1..260]	B#16#0
*1.0		BYTE	
=650.0		END_STRUCT	
=650.0		END_STRUCT	

server_client

TRUE:	S7 is server
FALSE:	S7 is client

single_write

In "S7 is client" mode, the function codes 5 and 6 are used with the single_write = TRUE parameter for write jobs with length 1. If single_write = FALSE, function codes 15 and 16 are used for all write jobs.

connect_at_startup

Specifies the time of connection establishment. If connect_at_startup is set to TRUE, the connection is established – independently of ENQ_ENR – immediately after CPU restart. In this case, a data job may only be transmitted if the connection was established correctly (CONN_ESTABLISHED = TRUE) or if a corresponding error is displayed at ERROR and STATUS. The status outputs are updated not later than when CONN_TIMEOUT expires.

FALSE:	Connection establishment with set ENQ_ENR
TRUE:	Connection establishment immediately after restart

Eight data areas

There are eight data areas available for mapping the MODBUS addresses in the S7 memory. At least the first data area must be defined; the remaining seven data areas are optional. Depending on the type of job, data is either read from or written to the data areas.

You can only read from one DB or write to one DB with any job. Access to registers or bit values that are located in several DBs, even if the numbers are consecutive without gaps, are to be divided into two jobs. Keep this in mind during configuration.

It is possible to map more Modbus areas (registers or bit values) in one data block than can be processed with one frame.

data_type_x

The data_type_x parameter specifies which MODBUS data types are mapped in this data block. If the value 0 is entered in data_type_x, the corresponding data area is not used.

Identifier	Data type	Data width
0	Area not used	
1	Coils	Bit
2	Inputs	Bit
3	Holding register	Word
4	Input register	Word

db_x

The db_x parameter specifies the data block in which MODBUS registers or bit values defined below are mapped. The DB number 0 is not permitted because it is reserved for the system.

Table 4- 7 db_x

DB number	1 to 65535 (W#16#0001 to W#16#FFFF)

The data block must be created 2 bytes longer than necessary for the configured data. The last two bytes are used for internal purposes.

start_x, end_x

start_x specifies the first Modbus address which is mapped in data word 0 of the DB. The end_x parameter defines the address of the last Modbus address.

The data word number in the S7 DB in which the last Modbus address input is entered is calculated according to the following formula for register access:

DBW number = (end_x – start_x) * 2

The data byte number in the S7 DB in which the last Modbus address input is entered is calculated according to the following formula for bit access:

DBB number = (end_x – start_x + 7) / 8

The defined data areas must not overlap. The end_x parameter must not be lower than start_x. In case of an error, startup of the FB is aborted with an error. If both values are identical, one Modbus address (1 register or 1 bit value) is assigned.

Section 6.3 contains an example of the mapping of the MODBUS addresses to S7 memory areas.

Table 4- 8 start_x, end_x

MODBUS address	0 to 65535 (W#16#0000 bis W#16#FFFF)

internal_send_buffer

This array is used internally in the FB for the send data. Access or changes to this area are not permitted.

internal_recv_buffer

This array is used internally in the FB for the receive data. Access or changes to this area are not permitted.

Licensing 5

General information

The MODBUSPN block must be licensed on each CPU individually. Licensing takes place in two steps: reading out the IDENT_CODE and entering the registration key REG_KEY. OB121 must exist on the CPU.

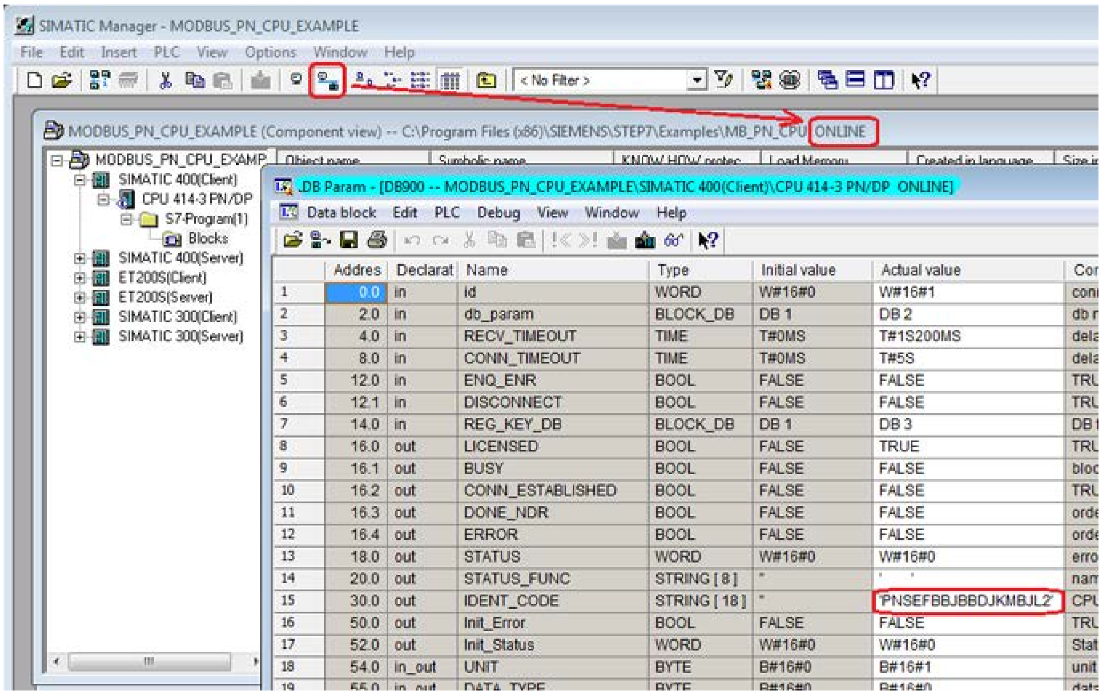
Reading out the IDENT_CODE

To read out the IDENT_CODE, follow the steps below:

1. Set the parameters for the MODBUSPN block according to your requirements in a cyclic OB (OB1 or cyclic interrupt OB).

   Load the program to the CPU and set it to RUN.

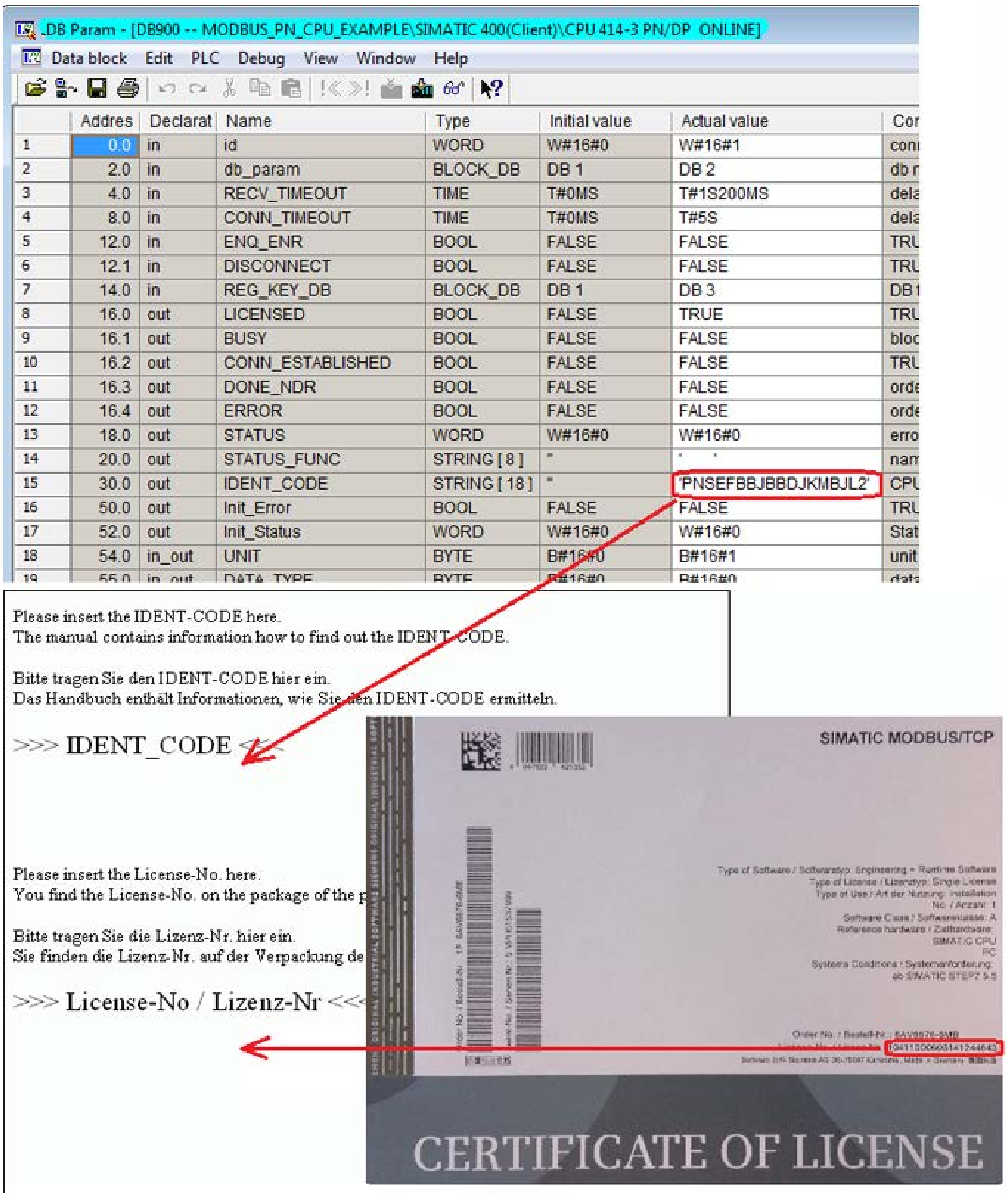
2. Open the project in online mode in the SIMATIC Manager. Open the instance DB of the Modbus block in this online project.


SIMATIC Manager - MODBUS_PN_CPU_EXAMPLE
File Edit Insert PLC View Options Window Help
< No Filter >
MODBUS_PN_CPU_EXAMPLE (Component view) -- C:\Program Files (x86)\SIEMENS\STEP7\Examples\MB_PN_CPU ONLINE
MODBUS_PN_CPU_EXAMP
SIMATIC 400(Client)
CPU 414-3 PN/DP
S7-Program(1)
Blocks
SIMATIC 400(Server)
ET200S(Client)
ET200S(Server)
SIMATIC 300(Client)
SIMATIC 300(Server)
Object name
Symbolic name
KNOW HOW protec
Load Memory
Created in language
Size i
.DB Param - [DB900 -- MODBUS_PN_CPU_EXAMPLE\SIMATIC 400(Client)\CPU 414-3 PN/DP ONLINE]
Data block Edit PLC Debug View Window Help

	Addres	Declarat	Name	Type	Initial value	Actual value	Cor
1	0.0	in	id	WORD	W#16#0	W#16#1	con
2	2.0	in	db_param	BLOCK_DB	DB 1	DB 2	db r
3	4.0	in	RECV_TIMEOUT	TIME	T#0MS	T#1S200MS	dela
4	8.0	in	CONN_TIMEOUT	TIME	T#0MS	T#5S	dela
5	12.0	in	ENQ_ENR	BOOL	FALSE	FALSE	TRU
6	12.1	in	DISCONNECT	BOOL	FALSE	FALSE	TRU
7	14.0	in	REG_KEY_DB	BLOCK_DB	DB 1	DB 3	DB
8	16.0	out	LICENSED	BOOL	FALSE	TRUE	TRU
9	16.1	out	BUSY	BOOL	FALSE	FALSE	bloc
10	16.2	out	CONN_ESTABLISHED	BOOL	FALSE	FALSE	TRU
11	16.3	out	DONE_NDR	BOOL	FALSE	FALSE	orde
12	16.4	out	ERROR	BOOL	FALSE	FALSE	orde
13	18.0	out	STATUS	WORD	W#16#0	W#16#0	erro
14	20.0	out	STATUS_FUNC	STRING [8]	''	' '	nam
15	30.0	out	IDENT_CODE	STRING [18]	''	'PNSEFBBJBBDJKMBJL2'	CPU
16	50.0	out	Init_Error	BOOL	FALSE	FALSE	TRU
17	52.0	out	Init_Status	WORD	W#16#0	W#16#0	Stat
18	54.0	in_out	UNIT	BYTE	B#16#0	B#16#1	unit
19	55.0	in_out	DATA_TYPE	BYTE	B#16#0	B#16#0	data

3. An 18-character string is displayed at the IDENT_CODE output.

   Copy this string from the DB and paste it in the form SOFTWARE REGISTRATION FORM. This form is saved in the library path ..\Program Files\Siemens\Step7\S7LIBS\Modbus_PN_CPU during installation and is also available on the installation CD.
   Enter the license number from the product packaging in the form.


.DB Param - [DB900 -- MODBUS_PN_CPU_EXAMPLE\SIMATIC 400(Client)\CPU 414-3 PN/DP ONLINE]
Data block Edit PLC Debug View Window Help

	Addres	Declarat	Name	Type	Initial value	Actual value	Cor
1	0.0	in	id	WORD	W#16#0	W#16#1	con
2	2.0	in	db_param	BLOCK_DB	DB 1	DB 2	db r
3	4.0	in	RECV_TIMEOUT	TIME	T#0MS	T#1S200MS	dela
4	8.0	in	CONN_TIMEOUT	TIME	T#0MS	T#5S	dela
5	12.0	in	ENQ_ENR	BOOL	FALSE	FALSE	TRU
6	12.1	in	DISCONNECT	BOOL	FALSE	FALSE	TRU
7	14.0	in	REG_KEY_DB	BLOCK_DB	DB 1	DB 3	DB 1
8	16.0	out	LICENSED	BOOL	FALSE	TRUE	TRU
9	16.1	out	BUSY	BOOL	FALSE	FALSE	bloc
10	16.2	out	CONN_ESTABLISHED	BOOL	FALSE	FALSE	TRU
11	16.3	out	DONE_NDR	BOOL	FALSE	FALSE	orde
12	16.4	out	ERROR	BOOL	FALSE	FALSE	orde
13	18.0	out	STATUS	WORD	W#16#0	W#16#0	erro
14	20.0	out	STATUS_FUNC	STRING [8]	''	' '	nam
15	30.0	out	IDENT_CODE	STRING [18]	''	'PNSEFBBJBBDJKMBJL2'	CPU
16	50.0	out	Init_Error	BOOL	FALSE	FALSE	TRU
17	52.0	out	Init_Status	WORD	W#16#0	W#16#0	Stat
18	54.0	in_out	UNIT	BYTE	B#16#0	B#16#1	unit
19	55.0	in_out	DATA_TYPE	BYTE	B#16#0	B#16#0	data

Please insert the IDENT-CODE here.
The manual contains information how to find out the IDENT-CODE.

Bitte tragen Sie den IDENT-CODE hier ein.
Das Handbuch enthält Informationen, wie Sie den IDENT-CODE ermitteln.

>>> IDENT_CODE <<<

Please insert the License-No. here.
You find the License-No. on the package of the p

Bitte tragen Sie die Lizenz-Nr. hier ein.
Sie finden die Lizenz-Nr. auf der Verpackung de

>>> License-No / Lizenz-Nr <<

SIMATIC MODBUS/TCP
Type of Software / Softwaretyp: Engineering + Runtime Software
Type of License / Lizenztyp: Single License
Type of Use / Art der Nutzung: Installation
No. / Anzahl: 1
Software Class / Softwareklasse: A
Reference hardware / Zielhardware:
SIMATIC CPU
PC
Systems Conditions / Systemanforderung:
ab SIMATIC STEP7 5.5
CERTIFICATE OF LICENSE

4. Send the form as Support Request (https://support.industry.siemens.com/my/ww/en/requests/#createRequest) to Customer Support. You will then receive the registration key for your CPU.
5. Note on the use in CFC: Online, the CFC editor can only display a certain number of characters. The full IDENT_CODE is displayed in the tooltip of the output parameter or in the IDB.

Entering the registration key REG_KEY

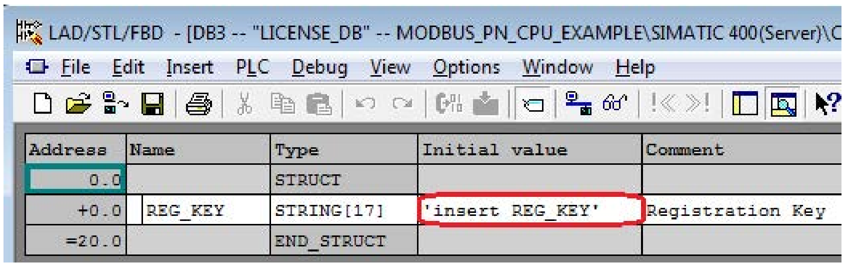
The registration code REG_KEY must be specified at each Modbus block call.

The REG_KEY must be saved in a global data block via which all Modbus blocks receive the required registration key.

Proceed as follows to enter the registration key REG_KEY:

1. Copy the ready-made licensing block DB3 from the "Modbus_PN_CPU" library to your project. If the DB number is already being used in the project, the license DB can also be renamed.
2. Open the license DB, copy the 17-character registration key and paste it into the "Initial value" column.

   Multiple keys can be input as list.


LAD/STL/FBD - [DB3 -- "LICENSE_DB" -- MODBUS_PN_CPU_EXAMPLE\SIMATIC 400(Server)\C
File Edit Insert PLC Debug View Options Window Help

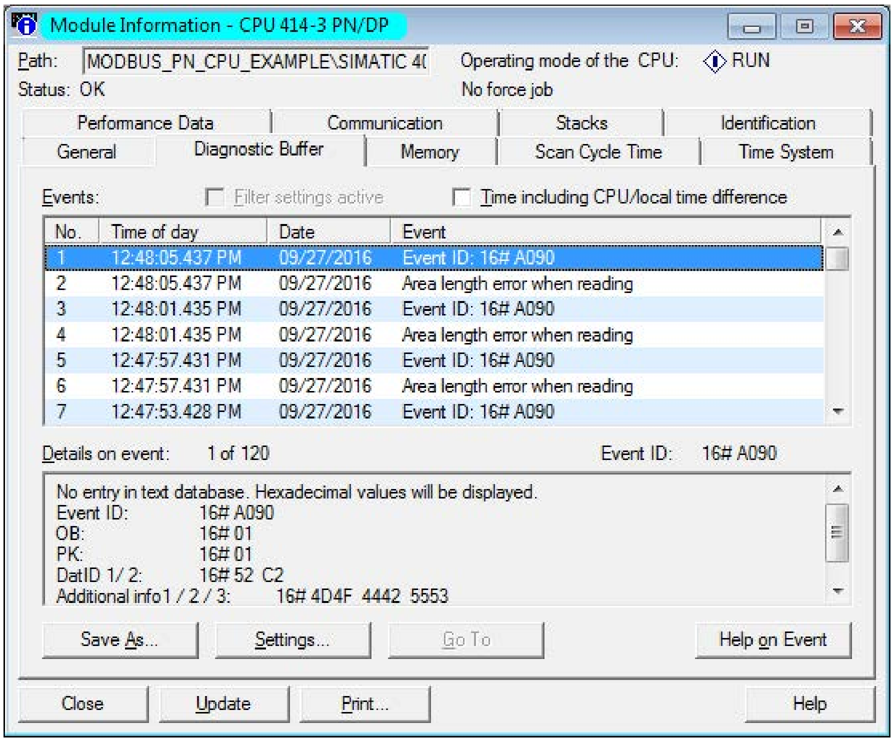
Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	REG_KEY	STRING[17]	'insert REG_KEY'	Registration Key
=20.0		END_STRUCT		

3. To avoid having to enter the registration key again after reloading the CPU, it needs to be entered permanently in the data block. Change to the data view of the DB with the menu item "View > Data View". With the menu command "Edit > Initialize Data Block", all the values from the "Initial value" column are transferred to the "Actual value" column.
4. In the cyclic OB, enter the data block number of the license DB at the REG_KEY parameter.
5. Download the modified blocks to the CPU. The registration key can be entered during runtime; a change from STOP -> RUN is not necessary.

The block is now licensed for this CPU.

Missing or incorrect licensing

If you enter an incorrect registration key or none at all, the SF LED (for S7-300 and IM151-8) or INTF LED (for S7-400) of the CPU flashes and a cyclic entry is made in the diagnostics buffer regarding the missing license. The error number for a missing license is W#16#A090.



Path: MODBUS_PN_CPU_EXAMPLE\SIMATIC 4(
Status: OK
Operating mode of the CPU: RUN
No force job

Events: ☐ Filter settings active ☐ Time including CPU/local time difference

No.	Time of day	Date	Event
1	12:48:05.437 PM	09/27/2016	Event ID: 16# A090
2	12:48:05.437 PM	09/27/2016	Area length error when reading
3	12:48:01.435 PM	09/27/2016	Event ID: 16# A090
4	12:48:01.435 PM	09/27/2016	Area length error when reading
5	12:47:57.431 PM	09/27/2016	Event ID: 16# A090
6	12:47:57.431 PM	09/27/2016	Area length error when reading
7	12:47:53.428 PM	09/27/2016	Event ID: 16# A090

Details on event: 1 of 120 Event ID: 16# A090

```
No entry in text database. Hexadecimal values will be displayed.
Event ID:         16# A090
OB:               16# 01
PK:               16# 01
DatID 1/ 2:       16# 52  C2
Additional info1 / 2 / 3:    16# 4D4F  4442  5553
```

WARNING

Inadvertent STOP state

If OB121 is missing in the controller, the CPU is set to STOP.

In the case of a missing or incorrect registration key, the Modbus communication is processed but W#16#A090 "No valid license available" is permantently displayed at the STATUS output.

6 MODBUSPN function block

6.1 Mode of operation of FB

General information

This MODBUSPN function block enables communication between a CPU with integrated PN interface and a partner which supports the Modbus/TCP protocol.

The function codes 1, 2, 3, 4, 5, 6, 15 and 16 are supported. Depending on the parameter assignment, the FB can be operated both as client and server.

The MODBUSPN block internally calls the MOD_CLI (FB901) and MOD_SERV (FB903) blocks. The MOD_CLI block contains the functionality of the Modbus client, the MOD_SERV block implements the functionality of the Modbus server.

Tasks of the FB

The function block fulfils the following tasks:

- Connection handling and data handling using T-blocks from the standard library
- Generating Modbus-specific request header when sending
- Checking the MODBUS-specific request header when receiving
- Checking whether the addressed data areas exist
- Generating exception frames if an error has occurred (only when S7 is server)

Exception code	Meaning
1	The sent function code is not supported.
2	There was access to an address that is non-existent or is not permitted.
3	An invalid length was specified for this function code.

- Data transfer from/to the set DB
- Monitoring the time it takes to establish and terminate the connection as well as to receive the data
- Maintaining the transaction identifier TI
- License check

Online Help

For the MODBUSPN function block, a block online help is available in the SIMATIC Manager. By selecting the block and pressing the "F1" key, the online help with the most important information on the module opens.

Calling the FB

For the program to run correctly, the MODBUSPN function block must be installed in a cyclic OB (OB1 or a time-driven OB, for example OB35).

The other MOD_CLI and MOD_SERV FBs contained in the library are called at a lower level and must not be called additionally in an OB.

The simultaneous call of the MODBUSPN in OB1 and in a time-driven OB (e.g. OB35) is not permitted.

OB121 must exist in the CPU. You can find more detailed information on this in the "Licensing" section.

Initialization of the FB

The MODBUSPN function block is initialized with a positive edge at the "Init" input.

- The initialization parameters must be assigned according to the plant configuration.
- The initialization parameters are applied to the instance DB.
- The runtime parameters are not evaluated during the initialization.
- The data from MODBUS_PARAM are checked for plausibility.

If a positive edge is detected at the "Init" parameter, the actions described above are carried out. If it was possible to complete the check without error, "Init" is reset, "Init_Error" and "Init_Status" display 0.

If errors occurred during the check, this is displayed at the "Init_Error" and "Init_Status" outputs. As long as an Init error is pending, no Modbus/TCP communication is possible via the block. The Init error must be corrected first.

Cyclic operation of the FB

In cyclic operation, MODBUSPN is called, for example, in OB35.

- The block functions are activated based on the runtime parameters.
- Changes to the runtime parameters are not evaluated while a job is in progress.
- The initialization parameters are not evaluated as long as no initialization has been carried out.

Programming error OB121

If the Modbus block is not yet licensed for this CPU, OB121 is called.

⚠ WARNING
Inadvertent STOP state
If OB121 is missing in the controller, the CPU is set to STOP.

Connection processing

The Modbus client performs the active establishment of the connection. The data is read out from the connection parameters in the MODBUS_PARAM DB.

A parameter in the connection parameter block (active_est) specifies whether the PN CPU is to serve as active or passive communication partner.
A communication channel to the link partner is opened during runtime for both connection types, active and passive, with the TCON function.

The time of the connection establishment is specified with a parameter in the MODBUS_PARAM DB (connect_at_startup).

The connection is terminated with the DISCONNECT parameter at the MODBUSPN FB.

Job initiation, S7 is client

A positive edge at the trigger input ENQ_ENR initiates a job. Depending on the input parameters UNIT, DATA_TYPE, START_ADDRESS, LENGTH and WRITE_READ, a MODBUS request frame is generated and sent to the partner station via the TCP/IP connection. The block waits for a response from the server for the set RECV_TIMEOUT time. If there is a timeout (no response from the server), the activated job is ended with an error. A new job can be initiated.

When a response frame is received, a plausibility check is performed. If the check returns a positive result, the required actions are performed, the job is ended without error and the output DONE_NDR is set. If errors were detected during the check, the job is ended with error, the ERROR bit is set and an error number is displayed at STATUS.

Activation of the FB, S7 is server

With a positive level at the ENQ_ENR trigger input, the FB is ready to receive a request frame from the client. The server is passive in this case and waits for a frame from the client. The received frame is checked. If the check returns a positive result, the request frame is answered. The user is informed about the completed frame traffic by the setting of the DONE_NDR bit. At this time, the executed function is indicated at the outputs UNIT, DATA_TYPE, START_ADDRESS, LENGTH and WRITE_READ.

An incorrect request frame results in an error message. The ERROR bit is set, the error number is indicated in STATUS and the request from the client is not processed. An exception frame is sent to the client, depending on the error.

6.2 Parameters of the MODBUSPN function block

Parameter	Decl.	Type	Description	Range of values	Init
id	IN	WORD	Connection ID according to the "id" configuration in the parameter DB	1 to 4095 W#16#1 to W#16#FFF	Yes
db_param	IN	BLOCK_DB	Parameter DB, contains all connection and Modbus data for this Modbus block instance	Depends on CPU	Yes
REG_KEY_DB	IN	BLOCK_DB	Data block with the registration key for licensing	Depends on CPU	No
RECV_TIMEOUT	IN	TIME	Monitoring time for data reception, min. 20 ms	T#20ms to T#+24d20h31m23s	No
CONN_TIMEOUT	IN	TIME	Monitoring time for establishing or terminating the connection, min. 100 ms	T#100ms to T#+24d20h31m23s	No
DISCONNECT	IN	BOOL	S7 is client: TRUE: The connection is terminated once the response frame has been received. S7 is server: TRUE: The connection is terminated if ENQ_ENR = FALSE.	TRUE/FALSE	No
ENQ_ENR	IN	BOOL	S7 is client: Job triggered on positive edge S7 is server: Ready to receive at positive level	TRUE/FALSE	No
LICENSED	OUT	BOOL	License status of the block: Block is licensed Block is not licensed	 TRUE FALSE	No
CONN_ ESTABLISHED	OUT	BOOL	TRUE: Connection is established FALSE: Connection is terminated	TRUE/FALSE	No
BUSY	OUT	BOOL	Processing status of T functions (TCON, TDISCON, TSEND or TRCV) Being processed Not being processed	 TRUE FALSE	No
DONE_NDR	OUT	BOOL	S7 is client: TRUE: Activated job completed without errors S7 is server: TRUE: Request from client was executed and reply was sent	TRUE/FALSE	No
ERROR	OUT	BOOL	TRUE: An error has occurred.	TRUE/FALSE	No

Parameter	Decl.	Type	Description	Range of values	Init
STATUS	OUT	WORD	Error number or status information	0 to FFFF	No
STATUS_FUNC	OUT	STRING[8]	Name of the function that caused the error at STATUS	Character	No
IDENT_CODE	OUT	STRING [18]	Identification for licensing. Request the license with this identification string.	Character	No
Init_Error	OUT	BOOL	TRUE: An error occurred during initialization.	TRUE/FALSE	No
Init_Status	OUT	WORD	Status of initialization	0 to FFFF	No
UNIT	IN/OUT	BYTE	Unit Identifier (INPUT for CLIENT function, OUTPUT for SERVER function)	0 to 255 B#16#0 to B#16#FF	No
DATA_TYPE	IN/OUT	BYTE	Data type to be processed (INPUT for CLIENT function, OUTPUT for SERVER function)		No
			Coils	1	
			Inputs	2	
			Holding register	3	
			Input register	4	
START_ ADDRESS	IN/OUT	WORD	MODBUS start address (INPUT for CLIENT function, OUTPUT for SERVER function)	0 to 65535 W#16#0000 to W#16#FFFF	No
LENGTH	IN/OUT	WORD	Number of values to be processed (INPUT for CLIENT function, OUTPUT for SERVER function)		No
			Coils		
			Read function	1 to 2000	
			Write function	1 to 1968	
			Inputs		
			Read function	1 to 2000	
			Holding register		
			Read function	1 to 125	
			Write function	1 to 123	
			Input register		
			Read function	1 to 125	
WRITE_ READ	IN/OUT	BOOL	Write access Read access (INPUT for CLIENT function, OUTPUT for SERVER function)	TRUE FALSE	No
Init	IN/OUT	BOOL	Initialization on a positive edge	TRUE/FALSE	No

General information

The parameters of the MODBUSPN FB are divided into two groups:

- Initialization parameters (written in lowercase)
- Runtime parameters (written in uppercase)

The initialization parameters are only evaluated and entered in the instance DB if there is a positive edge at the "Init" parameter. The initialization parameters are identified with "Yes" in the "Init" column in the above table.

A change to the initialization parameters during operation has no effect. After changing these parameters, for example, in test mode, the instance DB (I-DB) needs to be re-initialized with a positive edge at the "Init" parameter.

Runtime parameters can be changed during cyclic operation. It does not make any sense to change input parameters while a job is running. Preparations for the next job and the associated changes to the parameters should only start after the previous job was ended with DONE_NDR or ERROR.

In the "S7 is server" mode, the output parameters may only be evaluated when DONE_NDR is set.

The output parameters are displayed dynamically and are therefore only pending for 1 CPU cycle. This means they have to be copied to other memory areas for further processing or display in the variable table.

Ranges of values

With the ranges of values of the various parameters, CPU-specific restrictions may need to be taken into account.

id

A connection ID is required for each connection from the PN CPU to a communication partner. A different connection ID is to be used for each logical connection in the case of multiple communication partners. This connection ID is configured in the connection parameter block contained in the MODBUS_PARAM parameter data block. The connection ID uniquely describes the connection from the CPU to the link partner and can have the values 1 to 4095.

The connection ID from the connection parameter block must be entered here and must be unique throughout the entire CPU.

db_param

The db_param parameter contains the number of the MODBUS_PARAM data block. The connection-specific and Modbus-specific parameters which are required for communication between the PN CPU and the link partner are stored in this parameter data block.

The range of values for this parameter depends on the CPU. The DB number 0 is not permitted because it is reserved for the system. The DB number is entered in plain text as "DBxy".

If you want to implement several connections, the parameter data block can contain the necessary parameters of all connections in sequence. You can also create a separate parameter data block for each connection.

REG_KEY_DB

The block must be licensed on every CPU. With correct entry of the registration key, the block is licensed and Modbus communication can be used without restrictions. The number of the data block which contains the registration key is specified. It is possible to enter several registration keys one under the other in the DB. The Modbus block searches the DB for the suitable registration key. You can find additional information in the section "Licensing".

RECV_TIMEOUT

The monitoring time RECV_TIMEOUT monitors the reception of the response frame from the link partner. The minimum value is 20 ms.

If the RECV_TIMEOUT is set to < 20 ms in "S7 is client" operating mode, a corresponding error message is displayed and the active job is rejected. When the monitoring time has elapsed, the activated job is ended with an error.

If the RECV_TIMEOUT is set to < 20 ms in "S7 is server" operating mode, the default value of 1.2 s is used. If the monitoring time is exceeded, an error is reported. The RECV_TIMEOUT monitors the runtime of the TCP stream. The break between individual client requests is not taken into consideration.

CONN_TIMEOUT

The connection establishment or termination is monitored with the CONN_TIMEOUT time. The minimum value is 100 ms.

If the connection could not be established or terminated successfully within the configured monitoring time, a corresponding error message is displayed at the STATUS output.

In "S7 is client" mode, a CONN_TIMEOUT that is configured too low is set to 5 s when connect_at_startup = TRUE. An error message is output and the activated job rejected in cyclic operation if the CONN_TIMEOUT is too short.

The default value of 5 s is used if the CONN_TIMEOUT was set to < 100 ms in "S7 is server" operating mode.

DISCONNECT

With DISCONNECT = TRUE in "S7 is client" mode, it is specified that the connection is to be terminated after the data transfer. With DISCONNECT = TRUE in "S7 is server" mode, the connection is terminated when the ENQ_ENR parameter is set to FALSE.

The parameter is a runtime parameter and can be set or reset as required.

ENQ_ENR

"S7 is client" operating mode: The data transfer is initiated on a positive edge. The request is generated with the input parameters UNIT, DATA_TYPE, START_ADDRESS, LENGTH and WRITE_READ. A new job can only be sent if the previous job was completed with DONE_NDR or ERROR.

If the connection is not established (CONN_ESTABLISHED = FALSE), the connection is established first and the data transfer subsequently executed.

"S7 is server" operating mode: With a positive level at the input, the FB is activated. Frames can be received from the client. If the connection is not terminated when ENQ_ENR is set (CONN_ESTABLISHED = FALSE), the termination of the connection is activated.
If ENQ_ENR changes from TRUE to FALSE during operation, the connection is possibly terminated depending on the setting at the DISCONNECT parameter.
When the ENQ_ENR input is not set and a connection is established, the received data is discarded.

LICENSED

If this output is set to TRUE, the MODBUS block is licensed on this CPU. If the output has the status FALSE, no or an incorrect license string was entered. You can find additional information in the section "Licensing".

CONN_ESTABLISHED

CONN_ESTABLISHED = TRUE indicates that there is a connection to the link partner and data can be transferred.

When CONN_ESTABLISHED = FALSE, there is no connection to the link partner.

BUSY

If this output is set, one of the T functions TCON, TDISCON, TSEND or TRCV is being processed.

DONE_NDR

In "S7 is client" mode, the active job was completed without errors. With a read function, the response data from the server has already been entered in the DB; with a write function, the response to the request frame was received from the server.

In "S7 is server" mode, the output displays frame traffic with the client that has been completed without errors. The job parameters of the client are indicated in the UNIT, DATA_TYPE, START_ADDRESS, LENGTH and WRITE_READ parameters. These outputs are only valid as long as DONE_NDR is set.

ERROR

An error is detected when this output is set.

In "S7 is client" mode, the active job was completed with errors. The associated error number is indicated at the STATUS output.

In "S7 is server" mode, an error was detected in a request frame of the client or when sending the response frame. The associated error number is indicated at the STATUS output.

STATUS

When ERROR is set, the STATUS output shows the error number; when ERROR is not set, it shows status information.

The error numbers and status information are described in the "Diagnostics" section.

STATUS_FUNC

This parameter indicates the name of the function that caused the error in the form of a character string.

IDENT_CODE

After the CPU has started up, an 18-character identification code with which the REG_KEY (registration key) for Modbus communication is requested is indicated at this parameter.

You can find additional information in the section "Licensing".

Init_Error

If an error occurred during initialization, this is indicated by Init_Error = TRUE.

Init_Status

If Init_Error is set, the Init_Status output indicates the error number. The error numbers are described in the "Diagnostics" section.

UNIT

The UNIT parameter, Unit Identifier, refers to the unique assignment of the link partner. It is mainly necessary if there are several serial devices downstream of a converter which are addressed with different UNIT numbers.

In the "S7 is client" function, the UNIT parameter is an input parameter. This input needs to be set according to the requirements. The FB enters this value in the request frame and checks the value when it receives the response. It must be noted that some link partners expect a specific UNIT number.

In the "S7 is server" function, the UNIT parameter is an output parameter. The FB enters the value from the request frame in the response. When the job is completed, the output is set to the received value.

DATA_TYPE

The DATA_TYPE parameter indicates which MODBUS data type is processed with the current request. The following values are permitted:

Coils	B#16#1
Inputs	B#16#2
Holding register	B#16#3
Input register	B#16#4

In "S7 is client" mode, this is an input parameter; in "S7 is server" mode, this is an output parameter. The different data types are directly related to the used function codes.

Data type	DATA_TYPE	Function	Length	single_write	Function code
Coils	1	read	any	irrelevant	1
Coils	1	write	1	TRUE	5
Coils	1	write	1	FALSE	15
Coils	1	write	>1	irrelevant	15
Inputs	2	read	any	irrelevant	2
Holding register	3	read	any	irrelevant	3
Holding register	3	write	1	TRUE	6
Holding register	3	write	1	FALSE	16
Holding register	3	write	>1	irrelevant	16
Input register	4	read	any	irrelevant	4

START_ADDRESS

The START_ADDRESS parameter determines the first MODBUS address that is written or read.

In "S7 is client" mode, this is an input parameter; in "S7 is server" mode, this is an output parameter.

LENGTH

The LENGTH parameter determines the number of MODBUS values that are written or read.

With read functions, a maximum of 125 registers are possible per request for holding and input registers. For coils and inputs, a maximum of 2000 bits are possible. With write functions, the maximum number of registers is 123 for holding registers and 1968 bits for coils.
The registers or bit values processed with a request must be located within one DB.

In "S7 is client" mode, this is an input parameter; in "S7 is server" mode, this is an output parameter.

WRITE_READ

This parameter defines whether a reading or writing function is to be performed. If the input/output has the value FALSE, it is a read function. The value TRUE defines a write function.

Only holding registers and coils can be written to. Input registers and inputs can only be read. In "S7 is client" mode, this is an input parameter; in "S7 is server" mode, this is an output parameter.

Init

The Modbus block is initialized on a positive edge at the Init parameter. The initialization can only be performed when no job is currently running. This must be ensured with ENQ_ENR = FALSE and BUSY = FALSE in the program.

Note

With initialization, the configured connections are terminated and re-established. If the id parameter changes, the connections must be manually terminated prior to the initialization with DISCONNECT = TRUE.

6.3 Example of the address mapping

Interpretation of the MODBUS addresses

The MODBUS data model is based on a series of memory areas that have different characteristics. These memory areas are distinguished in some systems, for example, MODICON PLCs, by means of the register address or bit address. The holding register with offset 0, for example, is referred to as register 40001 (memory type 4xxxx, reference 0001).

This often causes confusion because some manuals describe and mean the register address of the application layer and others the register/bit address actually transferred in the protocol.

The MODBUS FB uses the actually transferred Modbus address in its start_x, end_x and START_ADDRESS parameters. This means that register/bit addresses from 0000 hex to FFFF hex can be transferred with each function code.

Example: Parameter assignment for the data areas

Data area 1	data_type_1	B#16#3	Holding register
	db_1	W#16#B	DB 11
	start_1	W#16#0	Start address: 0
	end_1	W#16#1F3	End address: 499
Data area 2	data_type_2	B#16#3	Holding register
	db_2	W#16#C	DB 12
	start_2	W#16#2D0	Start address: 720
	end_2	W#16#384	End address: 900
Data area 3	data_type_3	B#16#4	Input register
	db_3	B#16#D	DB 13
	start_3	W#16#2D0	Start address: 720
	end_3	W#16#3E8	End address: 1000
Data area 4	data_type_4	B#16#0	Not used
	db_4	0	0
	start_4	0	0
	end_4	0	0
Data area 5	data_type_5	B#16#1	Coils
	db_5	W#16#E	DB 14
	start_5	W#16#280	Start address: 640
	end_5	W#16#4E2	End address:1250
Data area 6	data_type_6	B#16#2	Inputs
	db_6	W#16#F	DB 15
	start_6	W#16#6A4	Start address:1700
	end_6	W#16#8FC	End address: 2300

Data area 7	data_type_7	B#16#1	Coils
	db_7	W#16#10	DB 16
	start_7	W#16#6A4	Start address: 1700
	end_7	W#16#8FC	End address: 2300
Data area 8	data_type_8	B#16#0	Not used
	db_8	0	0
	start_8	0	0
	end_8	0	0

For this example:

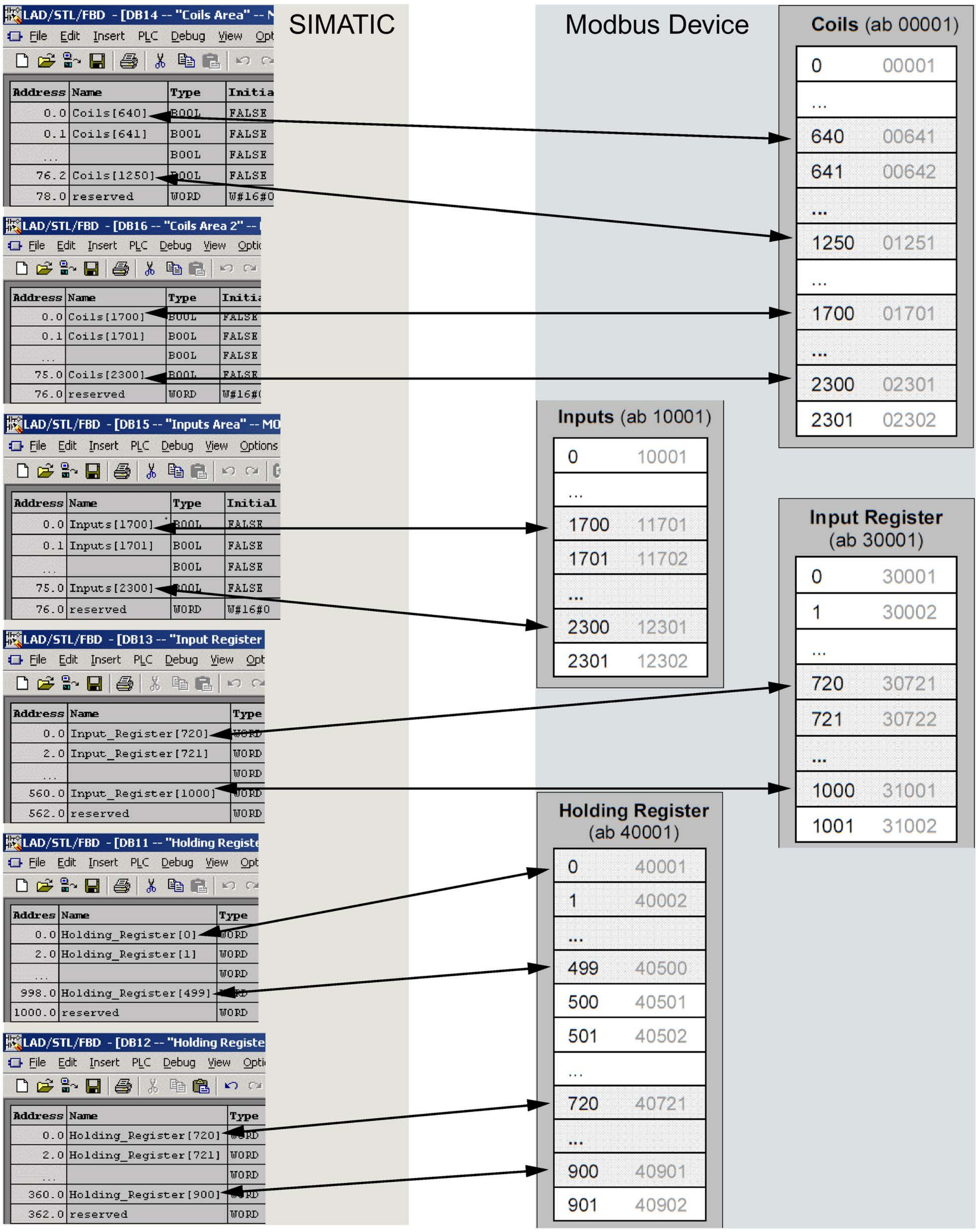
- DB11 has a size of 1002 bytes, a total of 500 registers are mapped (register 0 – register 499) + 2 reserved bytes.
- DB12 has a size of 364 bytes, a total of 181 registers are mapped (register 720 – register 900) + 2 reserved bytes.
- DB13 has a size of 564 bytes, a total of 281 registers are mapped (register 720 – register 1000) + 2 reserved bytes.
- DB14 has a size of 80 bytes, a total of 611 coils (bits) are mapped (coil 640 – coil 1250) + 2 reserved bytes.
- DB15 has a size of 78 bytes, a total of 601 inputs (bits) are mapped (input 1700 – input 2300) + 2 reserved bytes.
- DB16 has a size of 78 bytes, a total of 601 coils (bits) are mapped (coil 1700 – coil 2300) + 2 reserved bytes.

Address mapping

The figure below shows a comparison of the SIMATIC memory areas with the register-oriented and bit-oriented memory allocation of the Modbus devices. The figure references the parameter assignment described above.

In the Modbus device: The Modbus addresses shown in black relate to the data link layer, those shown in gray relate to the application layer.

In the SIMATIC: The SIMATIC addresses shown in the first column are the offset in the DB. The Modbus register numbers are entered in the square brackets.


SIMATIC
Modbus Device
LAD/STL/FBD - [DB14 -- "Coils Area" --
LAD/STL/FBD - [DB16 -- "Coils Area 2" --
LAD/STL/FBD - [DB15 -- "Inputs Area" -- MO
LAD/STL/FBD - [DB13 -- "Input Register
LAD/STL/FBD - [DB11 -- "Holding Registe
LAD/STL/FBD - [DB12 -- "Holding Registe
File Edit Insert PLC Debug View Options

Address	Name	Type	Initia
0.0	Coils[640]	BOOL	FALSE
0.1	Coils[641]	BOOL	FALSE
...		BOOL	FALSE
76.2	Coils[1250]	BOOL	FALSE
78.0	reserved	WORD	W#16#0

Address	Name	Type	Initia
0.0	Coils[1700]	BOOL	FALSE
0.1	Coils[1701]	BOOL	FALSE
...		BOOL	FALSE
75.0	Coils[2300]	BOOL	FALSE
76.0	reserved	WORD	W#16#0

Address	Name	Type	Initial
0.0	Inputs[1700]	BOOL	FALSE
0.1	Inputs[1701]	BOOL	FALSE
...		BOOL	FALSE
75.0	Inputs[2300]	BOOL	FALSE
76.0	reserved	WORD	W#16#0

Address	Name	Type
0.0	Input_Register[720]	WORD
2.0	Input_Register[721]	WORD
...		WORD
560.0	Input_Register[1000]	WORD
562.0	reserved	WORD

Addres	Name	Type
0.0	Holding_Register[0]	WORD
2.0	Holding_Register[1]	WORD
...		WORD
998.0	Holding_Register[499]	WORD
1000.0	reserved	WORD

Address	Name	Type
0.0	Holding_Register[720]	WORD
2.0	Holding_Register[721]	WORD
...		WORD
360.0	Holding_Register[900]	WORD
362.0	reserved	WORD

Coils (ab 00001)

0	00001
...	
640	00641
641	00642
...	
1250	01251
...	
1700	01701
...	
2300	02301
2301	02302

Inputs (ab 10001)

0	10001
...	
1700	11701
1701	11702
...	
2300	12301
2301	12302

Input Register (ab 30001)

0	30001
1	30002
...	
720	30721
721	30722
...	
1000	31001
1001	31002

Holding Register (ab 40001)

0	40001
1	40002
...	
499	40500
500	40501
501	40502
...	
720	40721
...	
900	40901
901	40902

6.4 Data and standard functions used by the FB

Instance DB

The MODBUSPN function block saves its data in an instance DB. This instance DB is generated by STEP 7 the first time the FB is called.

The instance data block contains parameter values of the type Input, Output, Input/Output and static variables which it requires to run. These variables are retentive and remain valid between FB calls. The internal execution of the FB is controlled by the variables.

Memory requirements of the instance DB:

Instance DB	Work memory	Load memory
MODBUSPN	Approx. 1.5 KB	Approx. 3 KB

Local variables

A maximum of 156 bytes of local data are required for an FB MODBUSPN call.

Parameter DB

The connection parameters and Modbus-specific parameters are saved in the MODBUS_PARAM parameter DB.

Timers

The function block does not use any timers

Memory bits

The function block does not use any memory bits.

Standard FBs for connection processing

The MOD_CLI and MOD_SERV blocks called in the MODBUSPN FB use the TCON and TDISCON blocks from the standard library to establish or terminate connections between the CPU and the communication partner.

Standard FBs for data transfer

The MOD_CLI and MOD_SERV blocks called in the MODBUSPN FB use the TSEND and TRCV blocks from the standard library to transfer data between the CPU and the communication partner.

MODBUSPN: SFCs for other functions

The MODBUSPN FB uses the following SFCs from the standard library:

- SFC20 "BLKMOV"
- SFC24 "TEST_DB"
- SFC51 "RDSYSST"
- SFC52 "WR_USMSG"

MOD_CLI/MOD_SERV: SFCs for other functions

The MOD_CLI/MOD_SERV FBs use the following SFCs and SFBs from the standard library:

- SFC20 "BLKMOV"
- SFC24 "TEST_DB"
- SFB4 "TON"

Additional information

The TI (Transaction Identifier) parameter is controlled internally by the MODBUSPN block in client mode and incremented by 1 with each new job.

The time in which a loss of connection can be detected is determined by the Keep Alive Time parameter. You will find this parameter in the properties of the CPU in HW Config.

6.5 Renaming / rewiring functions and function blocks

Objective

If the numbers of the standard functions are already being used in your project or the number range is reserved for other applications (for example, in CFC), you can rewire the internally called FB63, FB64, FB65 and FB66 function blocks.

The MODBUSPN block is BlockPrivacy protected. The internally called blocks FB901 MOD_CLI and FB903 MOD_SERV therefore cannot be rewired.

The system functions SFC20, SFC24, SFC51 and SFC52 and the system function block SFB4 also cannot be rewired.

Rewiring

To rewire for the FBs follow the steps below:

1. Call up information about the addresses used with "Options > Reference Data > Display".
2. Set the address priority in the object properties of the block folder to "Absolute value".
3. In the SIMATIC Manager, select the "Options > Rewire" function to rewire the addresses into the unused range.
4. To be able to continue using symbols in diagnostics tools, update the symbol table with the changes.

If you want to check the changes, select "Options >Reference Data >Display".

Additional blocks 7

7.1 Support in CFC

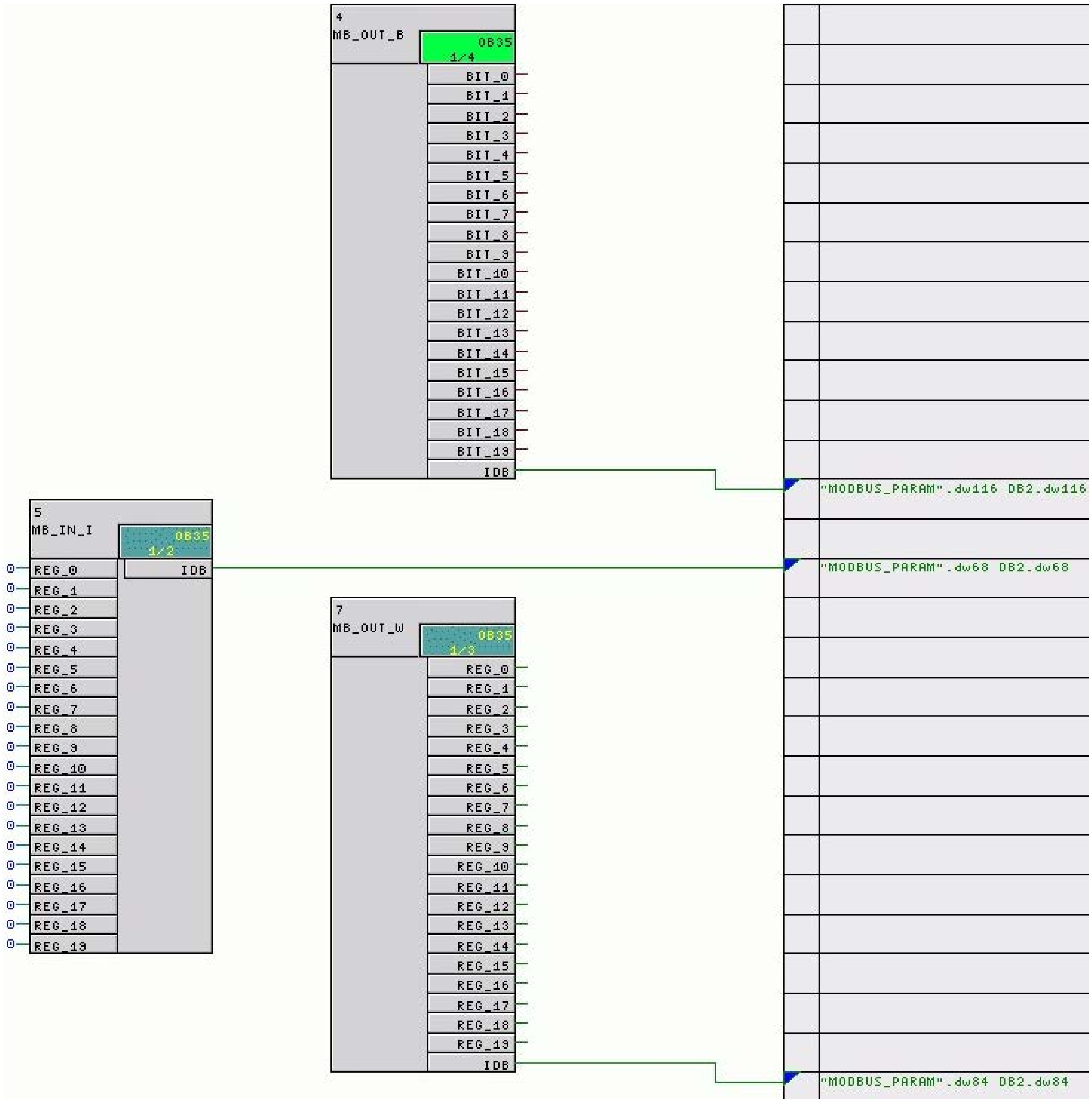
General

To support configuration in CFC, you have the option of configuring the Modbus values using "DataCollector FBs" instead of using global DBs. In this case, the send and receive buffers for the values are dragged to the CFC chart.

Application- example

The DataCollector FBs are placed in the CFC chart. The "IDB" output is connected to the DB parameters db_1 to db_8 in the parameter data block.

The Modbus values can subsequently be interconnected directly from the channel blocks to the DataCollector FB.



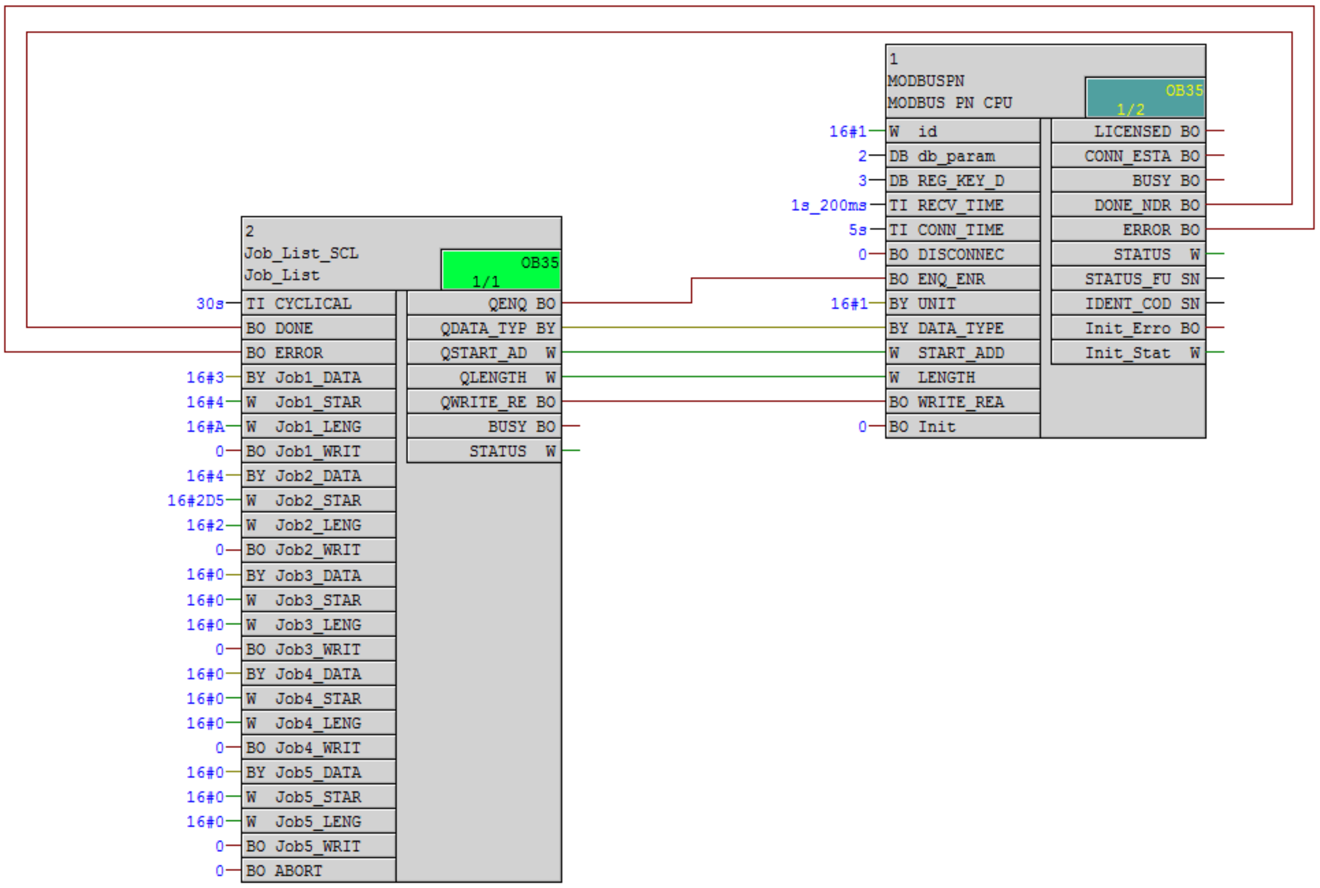
You can find the additional blocks and a detailed description here (http://support.automation.siemens.com/WW/view/en/62830463).

7.2 Job list for cyclic data exchange

General

With the Job_List block parameters can be set for a list of jobs that is worked through cyclically.

Application example



You can find the additional block and a detailed description here (http://support.automation.siemens.com/WW/view/en/62830463).

Diagnostics 8

Diagnostics functions

The diagnostics functions of the PN-CPU allow you to localize errors quickly. The following diagnostics options are available:

- Diagnostics with the display elements of the CPU
- Diagnostics with the STATUS outputs of the Modbus block

Display elements (LEDs)

The display elements indicate the operating status or possible error states of the CPU. The display elements give you an initial overview of any internal or external errors that have occurred as well as interface-specific errors.

STATUS outputs of the MODBUSPN

To allow error diagnostics, the MODBUSPN function block has STATUS outputs. By reading the STATUS outputs, you can obtain general information about errors that have occurred during communication. You can evaluate the STATUS parameters in the user program.

8.1 Diagnostics with the display elements of the CPU

Display functions

The display elements of the CPU provide information on the module status. The following display functions need to be distinguished:

Table 8- 1 Group error displays

PN-CPU 300 and IM 151-8/DP CPU	PN-CPU 400
SF group error If this LED flashes, the Modbus block is not yet licensed. You can find additional information in the section "Licensing".	INTF group error If this LED flashes, the Modbus block is not yet licensed. You can find additional information in the section "Licensing".

Table 8- 2 Special displays

PN-CPU 300 and IM 151-8 PN/DP	PN-CPU 400
RX/TX A frame is transferred via the interface.	

You will find a detailed description of the display elements in the respective CPU manual.

8.2 Diagnostics messages of the MODBUSPN FB

Messages at the STATUS outputs of the FB

The error messages are displayed at the status outputs of the MODBUSPN FB.

- The STATUS output displays error messages and status information during the block processing.
- The STATUS_FUNC output displays the name of the function that caused the error.
- The Init_Status output displays error messages and status information during the initialization.

Below is a listing of the error messages of specific to the FBs.

Error messages from the called SFCs and FBs

The Modbus FBs use the standard blocks SFC20, SFC24, SFC51 and SFC52. The error messages of these blocks are passed on unchanged to STATUS.

MOD_CLI and MOD_SERV use the standard blocks SFB4, FB63, FB64, FB65 and FB66. The error messages of these blocks are also passed on unchanged to STATUS.

You will find more information on these error messages in the diagnostics buffer or the online help for the SFCs/FCs in the SIMATIC Manager.

Table 8- 3 Error messages of the MODBUSPN FB to STATUS and Init_Status

Status (hex)	Event text	Remedy
A001	The MODBUS_PARAM parameter DB is too short or too long.	Correct the MODBUS_PARAM DB.
A002	The end_x parameter is lower than start_x.	Correct the information in the MODBUS_PARAM DB.
A003	A DB to which Modbus addresses are to be mapped is too short. Minimum length in bytes: - with registers: (end_x - start_x + 1)* 2 + 2 - with bit values: (end_x - start_x) / 8 + 1 + 2 Other possible causes: • S7 is client: Incorrect call parameters • S7 is server: incorrect address range in the request frame of the client. The S7 responds with an exception frame.	Lengthen the DB. • S7 is client: Correct the job parameter START_ADDRESS or LENGTH. • S7 is server: Change the client request.

Status (hex)	**Event text**		**Remedy**
A004	Only S7 is client: An invalid combination of DATA_TYPE and WRITE_READ was specified.		Correct the call parameters. Only data types 1 and 3 can be written.
A005	S7 is client: An invalid value was specified at the LENGTH parameter. S7 is server: The number of registers/bits in the request is invalid. The S7 responds with an exception frame.		S7 is client: Correct the LENGTH parameter. S7 is server: Change the number in the client's request frame.
	Ranges of values:		
	Read coils/inputs	1 to 2000	
	Write coils	1 to 1968	
	Read registers	1 to 125	
	Write holding registers	1 to 123	
A006	The area specified with DATA_TYPE, START_ADDRESS and LENGTH does not exist in data_type_1 to data_type_8. S7 is server: The S7 responds with an exception frame.		S7 is client: Correct the combination DATA_TYPE, START_ ADDRESS and LENGTH. S7 is server: Change the client request or correct the parameter assignment in the parameter DB.
A007	S7 is client: An invalid monitoring time was set for RECV_TIMEOUT or CONN_TIMEOUT. A value >= 20 ms must be entered for RECV_TIMEOUT, and a value >= 100 ms for CONN_TIMEOUT.		Correct the parameter assignment.
A009	S7 is client: The received transaction identifier TI is not the same as the one sent. The connection is terminated.		Record frames to check the data of the link partner.
A00A	S7 is client: The received UNIT is not the same as the one sent.		Record frames to check the data of the link partner.
A00B	S7 is client: The received function code is not the same as the one sent. S7 is server: An invalid function code was received. The S7 responds with an exception frame.		S7 is client: Record frames to check the data of the link partner. S7 is server: Change the client request. The Modbus FB processes the function codes 1, 2, 3, 4, 5, 6, 15 and 16.
A00C	The received bytecount does not match the number of registers. The connection is terminated.		Record frames to check the data of the link partner.
A00D	S7 is client: The register address/bit address or the number of registers/bits in the response is not the same as in the request.		Record frames to check the data of the link partner.

Status (hex)	Event text	Remedy
A00E	The length information in the Modbus-specific header does not match the specified number of registers/bits or the byte count in the request. The FB discards the data. The connection is terminated.	Record frames to check the data of the link partner
A00F	A protocol identifier other than 0 was received. The connection will be terminated.	Record frames to check the data of the link partner
A010	A DB number was assigned twice in the parameters db_1 to db_8 .	Correct the parameter assignment in the MODBUS_PARAM DB.
A011	An invalid value was specified for the DATA_TYPE input parameter (valid values are 1 - 4).	Correct the call parameters.
A012	The areas data_type_1 and data_type_2 set in the parameters overlap.	Correct the parameter assignment. The data areas must not contain any overlapping register areas.
A013	The areas data_type_1 and data_type_3 set in the parameters overlap.	
A014	The areas data_type_1 and data_type_4 set in the parameters overlap.	
A015	The areas data_type_1 and data_type_5 set in the parameters overlap.	
A016	The areas data_type_1 and data_type_6 set in the parameters overlap.	
A017	The areas data_type_1 and data_type_7 set in the parameters overlap.	
A018	The areas data_type_1 and data_type_8 set in the parameters overlap.	
A019	One of the db_x parameters was set to 0 even though the associated data_type_x is set to > 0. DB0 must not be used because this is reserved for the system.	Correct the parameter assignment for db_x to >0.
A01A	Incorrect length in the header: 3 to 253 bytes are permitted. The connection is terminated.	Record frames to check the data of the link partner.
A01B	S7 is server and function code 5: An invalid status was received for coil. The S7 responds with an exception frame.	Record frames to check the data of the link partner.
A023	The areas data_type_2 and data_type_3 set in the parameters overlap.	Correct the parameter assignment. The data areas must not contain any overlapping register areas.
A024	The areas data_type_2 and data_type_4 set in the parameters overlap.	
A025	The areas data_type_2 and data_type_5 set in the parameters overlap.	
A026	The areas data_type_2 and data_type_6 set in the parameters overlap.	
A027	The areas data_type_2 and data_type_7 set in the parameters overlap.	

Status (hex)	Event text	Remedy
A028	The areas data_type_2 and data_type_8 set in the parameters overlap.	
A034	The areas data_type_3 and data_type_4 set in the parameters overlap.	
A035	The areas data_type_3 and data_type_5 set in the parameters overlap.	
A036	The areas data_type_3 and data_type_6 set in the parameters overlap.	
A037	The areas data_type_3 and data_type_7 set in the parameters overlap.	
A038	The areas data_type_3 and data_type_8 set in the parameters overlap.	
A045	The areas data_type_4 and data_type_5 set in the parameters overlap.	
A046	The areas data_type_4 and data_type_6 set in the parameters overlap.	
A047	The areas data_type_4 and data_type_7 set in the parameters overlap.	
A048	The areas data_type_4 and data_type_8 set in the parameters overlap.	
A056	The areas data_type_5 and data_type_6 set in the parameters overlap.	
A057	The areas data_type_5 and data_type_7 set in the parameters overlap.	
A058	The areas data_type_5 and data_type_8 set in the parameters overlap.	
A067	The areas data_type_6 and data_type_7 set in the parameters overlap.	
A068	The areas data_type_6 and data_type_8 set in the parameters overlap.	
A078	The areas data_type_7 and data_type_8 set in the parameters overlap.	
A079	The connection ID specified for the id parameter is not contained in the MODBUS_PARAM parameter DB.	Correct the parameter assignment at the id input.
A07A	An invalid value was specified at the parameter id of the block (value range from 1 to 4095).	
A07B	The specified ID is contained twice in the parameter DB.	Correct the parameter assignment in the MODBUS_PARAM DB.
A07C	An invalid value was specified at the data_type_x parameter in the parameter DB (values from 0 to 4 are valid).	
A07D	The data_type_1 parameter has no entry in the parameter DB. The parameter area "_1" is the initial area and must be set.	
A07E	The number of the MODBUS_PARAM parameter DB or the number of the instance DB was specified by the MODBUSPN block at db_x.	

Status (hex)	Event text	Remedy
A07F	The DB specified at db_param is not a Modbus parameter DB. The length information in the DBW0 was changed or an incorrect DB was specified.	Correct the parameter assignment at the db_param input.
A080	The Modbus block has not yet been initialized.	The Modbus block must be initialized with Init = TRUE after the transfer of the IDB to the CPU.
A081	Only for S7 is client and function code 5: The data of the response is not the echo of the request.	Record frames to check the data of the link partner.
A082	Only for S7 is client and function code 6: The received register value is not the same as the one sent.	Record frames to check the data of the link partner.
A083	S7 is client: A job was triggered while the previous job is still in progress. The job will not be executed. This is status information. The ERROR bit is not set. There was an attempt to initialize the block while a job was still running or while ENQ_ENR was set.	Client: Only start a new job when the previous job ended with DONE_NDR = TRUE or ERROR = TRUE. Wait with the initialization until no job is running any more. Set ENQ_ENR = FALSE.
A084	No identification code IDENT_CODE could be determined for the licensing.	Please contact Product Support.
A085	An error occurred during license detection. The error is indicated with ERROR = TRUE at the first occurrence. It is indicated below as status message with ERROR = FALSE.	Check that there is no illegal write access to the license DB. The structure of REG_KEY must not be modified. If necessary, contact Product Support.
A086	An attempt was made to write to a write-protected data block.	Remove the write protection of the data block or use a different DB.
A090	The Modbus block has not yet been licensed for this CPU. This is status information. The ERROR bit is not set. Modbus communication runs even without license.	Read out the IDENT_CODE identification string for this CPU and use it to request the registration key (see Section 5 "Licensing").
A091	Only for S7 is client: An exception frame with exception code 1 was received as the reply.	The link partner does not support the requested function.
A092	Only for S7 is client: An exception frame with exception code 2 was received as the reply. There was access to a non-existent or illegal address on the link partner.	Correct LENGTH or START_ADDRESS in the FB call.

Status (hex)	Event text	Remedy
A093	Only for S7 is client: An exception frame with exception code 3 was received as the reply.	The link partner cannot process the received frame (for example, it does not support the requested length).
A094	Only for S7 is client: An exception frame with exception code 4 was received as the reply.	The link partner is in a status in which it cannot process the request.
A095	Only for S7 is client: An exception frame with an unknown exception code was received as the reply.	Check the error messages of the link partner and if necessary record the frames to check the data.
A100	The monitoring time CONN_TIMEOUT or RECV_TIMEOUT has expired for a job. The connection is terminated when RECV_TIMEOUT has expired.	Check the parameter assignment of the connection.
A101	The internal monitoring time of the TDISCON function has expired.	Contact Product Support.

8.3 Diagnostics messages of the linked in blocks

Diagnostics message

Table 8- 4 Error messages of the integrated FBs/SFCs at the STATUS output

STATUS (hex)	Event text	Remedy
7xxx	You will find more detailed information in the online help of the SIMATIC Manager.	Refer to the online help (SIMATIC Manager -> select block -> F1 key)
8xxx	You will find more detailed information in the online help of the SIMATIC Manager.	Refer to the online help (SIMATIC Manager -> select block -> F1 key)

8.4 Diagnostics messages of SFC24

Diagnostics message

Table 8- 5 Error messages of SFC24 at the STATUS output

STATUS (hex)	Event text	Remedy
80A1	DB number = 0 or too high for the CPU.	Select a permitted DB number.
80B1	The DB does not exist on the CPU.	All data blocks specified in db_x must be created and transferred to the CPU.
80B2	DB UNLINKED	Do not generate DB as UNLINKED.

Examples of applications 9

General

Two sample projects are stored in \Program Files\Siemens\Step7\Examples during the installation:

- One sample project in STL "MB_PN_CPU" and
- One sample project in CFC "MB_PN_CPU_CFC".

Simatic stations for all function variants are stored in the sample projects:

- SIMATIC station is client or server
- SIMATIC station is S7-300, S7-400 or IM 151-8 PN/DP CPU

Note

The S7 program is intended as a source of information and should not be considered as a binding solution proposal for a customer-specific plant configuration.

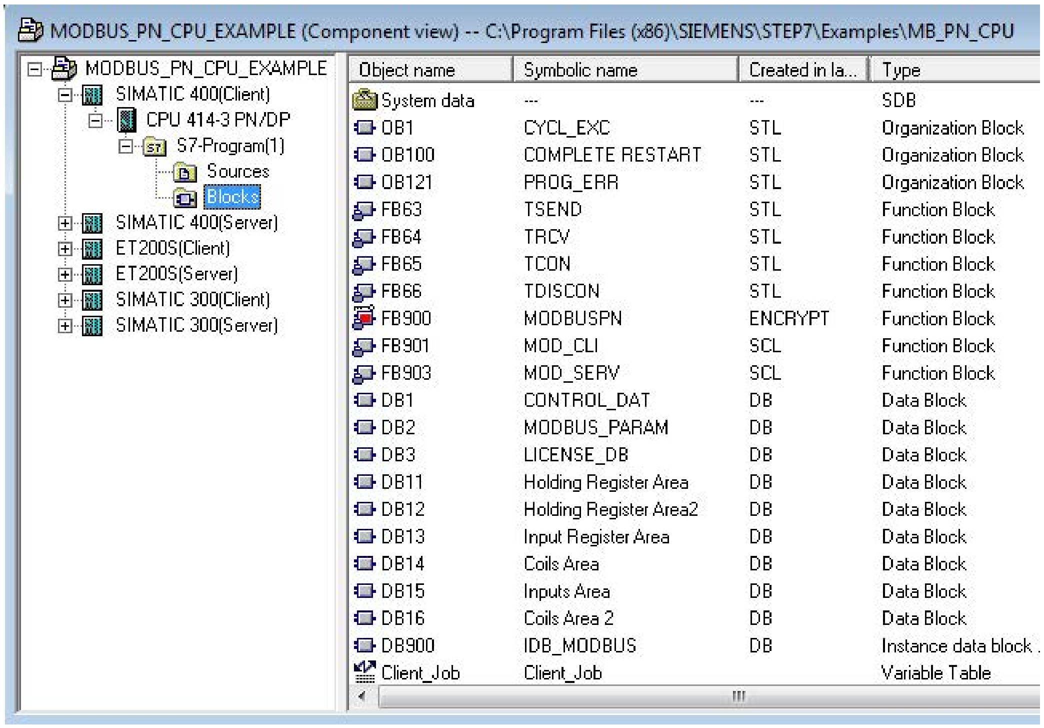
Program example

The program examples consist of the following blocks:

- Startup block OB100 with setting of the Init bit
- Programming error OB121
- Cyclic operation OB1 or OB35 with MODBUSPN call
- Global data blocks for starting a job (e.g. with the aid of a variables table) and for licensing
- Data blocks for register and bit values

9.1 Sample project in STL

Overview



MODBUS_PN_CPU_EXAMPLE (Component view) -- C:\Program Files (x86)\SIEMENS\STEP7\Examples\MB_PN_CPU

```
MODBUS_PN_CPU_EXAMPLE
  SIMATIC 400(Client)
    CPU 414-3 PN/DP
      S7-Program(1)
        Sources
        Blocks
  SIMATIC 400(Server)
  ET200S(Client)
  ET200S(Server)
  SIMATIC 300(Client)
  SIMATIC 300(Server)
```

Object name	Symbolic name	Created in la...	Type
System data	---	---	SDB
OB1	CYCL_EXC	STL	Organization Block
OB100	COMPLETE RESTART	STL	Organization Block
OB121	PROG_ERR	STL	Organization Block
FB63	TSEND	STL	Function Block
FB64	TRCV	STL	Function Block
FB65	TCON	STL	Function Block
FB66	TDISCON	STL	Function Block
FB900	MODBUSPN	ENCRYPT	Function Block
FB901	MOD_CLI	SCL	Function Block
FB903	MOD_SERV	SCL	Function Block
DB1	CONTROL_DAT	DB	Data Block
DB2	MODBUS_PARAM	DB	Data Block
DB3	LICENSE_DB	DB	Data Block
DB11	Holding Register Area	DB	Data Block
DB12	Holding Register Area2	DB	Data Block
DB13	Input Register Area	DB	Data Block
DB14	Coils Area	DB	Data Block
DB15	Inputs Area	DB	Data Block
DB16	Coils Area 2	DB	Data Block
DB900	IDB_MODBUS	DB	Instance data block .
Client_Job	Client_Job		Variable Table

Blocks used

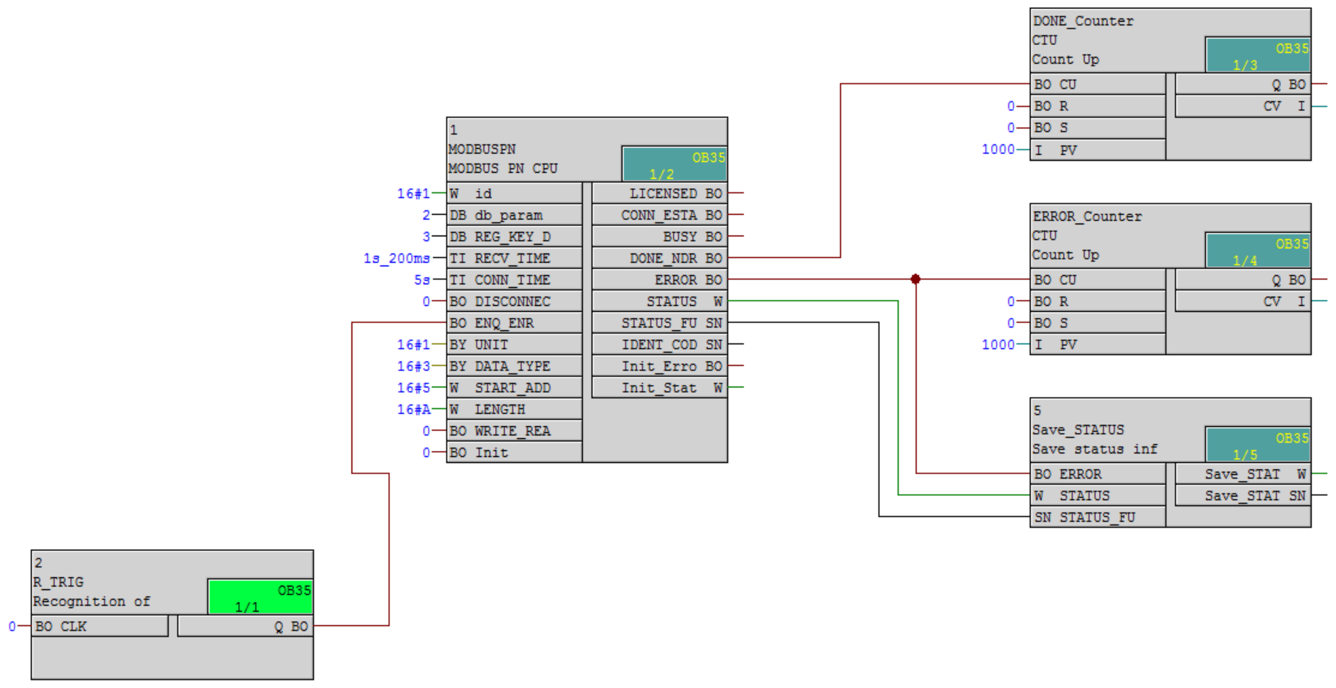
The block numbers are used in the supplied sample project.

Block	Symbol	Comment
OB 1	CYCL_EXC	Cyclic program execution
OB 100	COMPLETE RESTART	Startup OB
OB 121	PROG_ERR	Programming error OB
FB 900	MODBUSPN	User block FB MODBUSPN
FB 901	MOD_CLI	internally called FB MOD_CLI
FB 903	MOD_SERV	internally called FB MOD_SERV
DB 1	CONTROL_DAT	Work DB CONTROL DAT for MODBUSPN
DB 2	MODBUS_PARAM	Parameter DB for MODBUSPN
DB 3	LICENSE_DB	License DB for MODBUSPN
DB 11	Holding Register Area	Values DB for area 1
DB 12	Holding Register Area 2	Values DB for area 2
DB 13	Input Register Area	Values DB for area 3
DB 14	Coils Area	Values DB for area 5
DB 15	Inputs Area	Values DB for area 6
DB 16	Coils Area 2	Values DB for area 7
DB 900	IDB_MODBUS	Instance DB for MODBUSPN

9.2 Sample project in CFC

Overview

The sample project was created with CFC V8.0 Update 1.



Blocks used

The block numbers are used in the supplied sample project.

Block	Symbol	Comment
OB 35	CYCL_EXC	Cyclic program execution
OB 100	COMPLETE RESTART	Startup OB for the initialization
OB 121	PROG_ERR	Programming error OB
FB 99	Save_STATUS	Memory DB for errors
FB 900	MODBUSPN	User block MODBUSPN
FB 901	MOD_CLI	internally called FB MOD_CLI
FB 903	MOD_SERV	internally called FB MOD_SERV
DB 2	MODBUS_PARAM	Parameter DB for FB MODBUSPN
DB 3	LICENSE_DB	License DB for MODBUSPN
DB 11	Holding Register Area	Values DB for area 1
DB 12	Holding Register Area 2	Values DB for area 2
DB 13	Input Register Area	Values DB for area 3
DB 14	Coils Area	Values DB for area 5
DB 15	Inputs Area	Values DB for area 6
DB 16	Coils Area 2	Values DB for area 7

References A

The MODBUS Organization

MODBUS APPLICATION PROTOCOL SPECIFICATION
V1.1b3, April, 2012

Modbus home page (http://www.modbus.org)

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

Subject to change
A5E39465006-AB
© Siemens AG 2017

www.siemens.com/automation